

## KEYCLOAK

(

Ref :

<https://www.youtube.com/watch?v=6xaNsAClq0s&list=PLHXvj3cRjzs8TaT-RX1qJYYK2MjRro-P>

YouTube : [Dive Into Development](#) )

For Oauth Dependencies (to enable Oauth services)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
</dependency>
```

For Keycloak Dependencies:

Keycloak Version and Dependencies version need to be same

```
<!--This will be Deprecated soon in keycloak and In spring boot

https://mvnrepository.com/artifact/org.keycloak/keycloak-spring-boot-starter-->
<!--<dependency>
    <groupId>org.keycloak</groupId>
    <artifactId>keycloak-spring-boot-starter</artifactId>
    <version>23.0.4</version>
</dependency>-->
<!--

https://mvnrepository.com/artifact/org.keycloak/keycloak-servlet-filter-adapter-->
<!--<dependency>
    <groupId>org.keycloak</groupId>
    <artifactId>keycloak-servlet-filter-adapter</artifactId>
    <version>23.0.4</version>
    <scope>provided</scope>
</dependency>-->
<!--

https://mvnrepository.com/artifact/org.keycloak/keycloak-policy-enforcer-->
<dependency>
    <groupId>org.keycloak</groupId>
    <artifactId>keycloak-policy-enforcer</artifactId>
    <version>23.0.4</version>
</dependency>
```

```
<!--  
https://mvnrepository.com/artifact/org.keycloak/keycloak-core -->  
    <dependency>  
        <groupId>org.keycloak</groupId>  
        <artifactId>keycloak-core</artifactId>  
        <version>23.0.4</version>  
    </dependency>  
<!--  
  
https://mvnrepository.com/artifact/org.keycloak/keycloak-admin-client -->  
    <dependency>  
        <groupId>org.keycloak</groupId>  
        <artifactId>keycloak-admin-client</artifactId>  
        <version>23.0.4</version>  
    </dependency>
```

**To Store in your Own Database** Go to Keycloak Folder -> /keycloak/conf -> open keycloak.conf change the DB Details it will store DB automatically.

## Authentication & Authorization Flow in Various Scenarios

### Resource request:

It is used to send the request and needs to be authenticated by the authentication server and authentication token needs to be sent to the Resource server.

Eg: UI App,Postman,Client Server,Angular,Swagger UI.

### Authentication server:

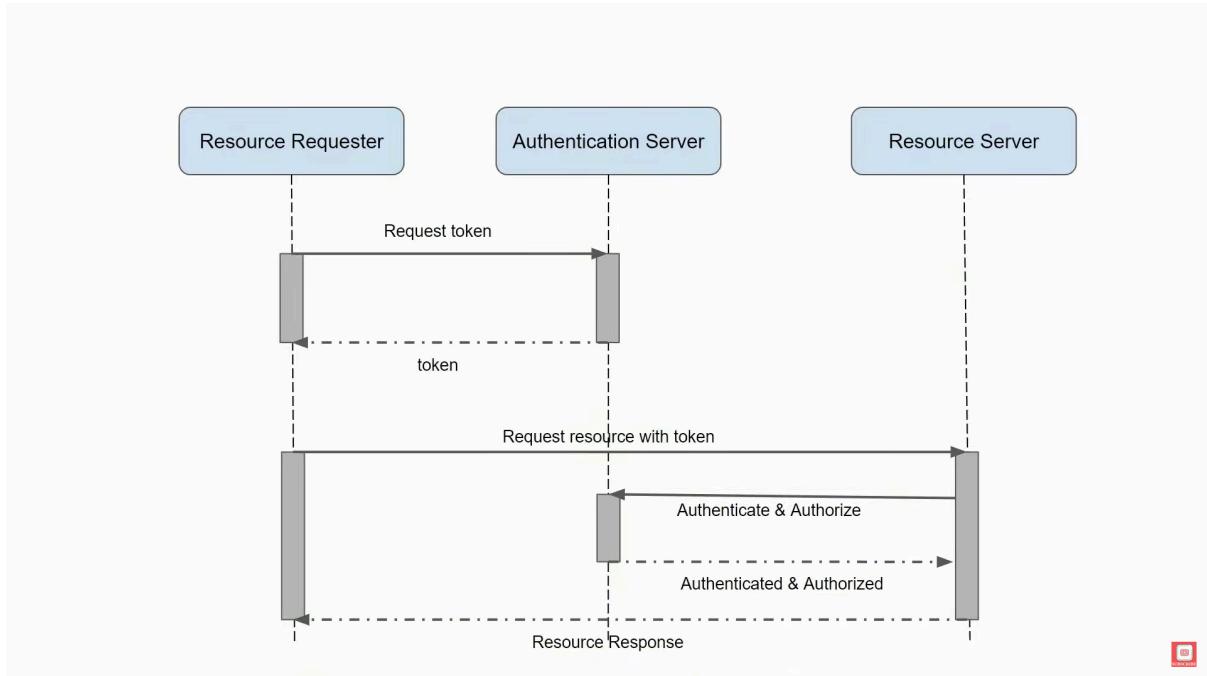
It will check the authenticate the resource request API and send it with Token generation,Refresh Token,Exp Date etc... and check if the token is valid or not for every request.

Eg: Keycloak,Okta,Asure and my own spring auth server.

### Resource Server:

It will contain the data and send a resource request.

Eg: our spring boot and Quarkus etc..



## KEYCLOAK CONFIG

Create Client ,Enter the name of client,click the create.

Client authentication → It will enable the credential key.

Authorization →

Standard Flow → redirect the login temp and redirect URL tab will be open.

Direct Access Flow → Mostly used for REST Api Calls.

Implicit Flow →

Server Account Flow →

OAuth 2.0 Device Authorization Flow →

OIDC CIBA Grant →

Getting Token:

Token\_endpoint : <http://localhost:8081/realm/quarkus-be/protocol/openid-connect/token>

```

client_id
client_secret
grant_type
username
password

```

Part 3 - Using keycloak APIs to generate token, check the validity, and logout the user.

POST /Token

http://127.0.0.1:8081/realm/div-dev/protocol/openid-connect/token

Key	Value	Description
client_id	quarkus-be	
client_secret	lgYLmLSZ9XrWKEZBslvdOBBgOnGvloAc	
grant_type	password	
username	sagar	
password	Demo@123	

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "error": "Realm does not exist"
3

```

Status: 200 OK Time: 58 ms Size: 2.76 KB Save as Example

To Check Token is Active or Not :

Introspection\_endpoint :

<http://localhost:8081/realm/quarkus-be/protocol/openid-connect/token/introspect>

token  
client\_id  
client\_secret

POST /Introspect Token

http://127.0.0.1:8081/realm/div-dev/protocol/openid-connect/token/introspect

Key	Value	Description
token	eyJhbGciOiJSU1NIiInR5C1gOIAiSlidUiwia2IkIA6ICJaU...	
client_id	quarkus-be	
client_secret	lgYLmLSZ9XrWKEZBslvdOBBgOnGvloAc	

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "exp": 1688427344,
3   "iat": 1688427844,
4   "jti": "693dc028-a88d-4d1c-86ae-7fcba5f26eaf",
5   "iss": "http://127.0.0.1:8081/realm/div-dev",
6   "aud": "account",
7   "sub": "1fa5c9e9-8a6e-4675-bfaf-f2d72257002f",
8   "typ": "Bearer",
9   "azp": "quarkus-be",
10  "session_state": "9239656f-ddbb-4660-ba9d-4a7a6ddd9bf2",
11  "name": "Sagar Kumar",
12  "given_name": "Sagar",
13  "family_name": "Kumar",
14  "preferred_username": "sagar",
15  "email": "sagar@demo.com",
16  "email_verified": true,

```

Status: 200 OK Time: 8 ms Size: 1 KB Save as Example

To Logout:

Logout\_endpoint : <http://localhost:8081/realm/quarkus-be/protocol/openid-connect/logout>

refresh\_token

client\_id

client\_secret

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a 'Collections' section with a 'Keycloak' folder containing three items: 'GET Token', 'GET Refresh Token', and 'GET Introspect Token'. Below these, there is a 'POST Logout' item. The main workspace area shows a 'POST' request to 'http://127.0.0.1:8081/realm/quarkus-be/protocol/openid-connect/logout'. The 'Body' tab is selected, showing the following parameters:

Key	Value	Description
refresh_token	eyJhbGciOiJIUzI1NlslnR5cClgOIAiSlUlwiia2lkIA6C1M2...	
client_id	quarkus-be	
client_secret	jqAsB4tVWXi8vOXJ0XPDVlcCb3WZMDj	

Below the body, there is a 'Response' section with a small illustration of an astronaut holding a rocket.

Refresh Token to generate valid Token:

Token API : <http://localhost:8081/realm/quarkus-be/protocol/openid-connect/token>

client\_id

client\_secret

grant\_type

refresh\_token

We can used to add Extra details in token using client scope

### Expanding token limit and refresh token limit :

For Refresh token : realm Setting -> session -> IN SSO Session Idle change it.

For Token : realm Setting -> Token -> In Access Token Lifespan change it.

**If Token is Expired Refresh Token can be Generated only 3 or 4 Time (which means we can set Limits)**

In realm Setting -> Token -> Revoke Refresh Token Enable it. It shows 0 which means it will be considered as 1.

## SPRING BOOT AND KEYCLOAK

### Authentication & Authorization In JWT & OPAQUE TOKEN

**Opaque tokens and JWT (JSON Web Tokens) are both methods used in authentication and authorization systems, but they serve slightly different purposes and have different characteristics. Here's a brief overview of each:**

#### 1. Opaque Tokens:

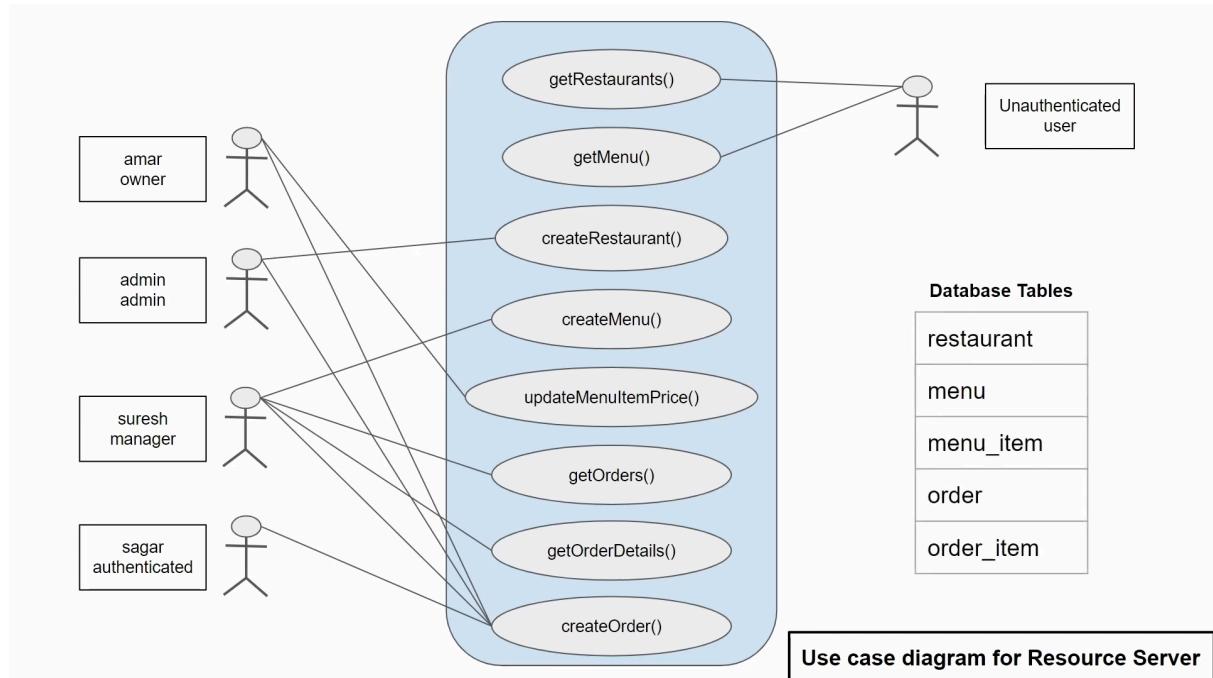
- An opaque token is typically a randomly generated string that serves as a reference to some server-side data.

- When a client sends an opaque token to a server, the server must validate the token by looking up its associated data in a database or some other storage mechanism.
- The client typically doesn't have any information about the contents of the token and cannot decode it or extract any meaningful information from it.
- Opaque tokens are generally used in scenarios where the server needs to maintain control over the data associated with the token and wants to minimize the amount of information sent over the network.
- They are commonly used in sessions-based authentication systems, where the server maintains session state on behalf of the client.

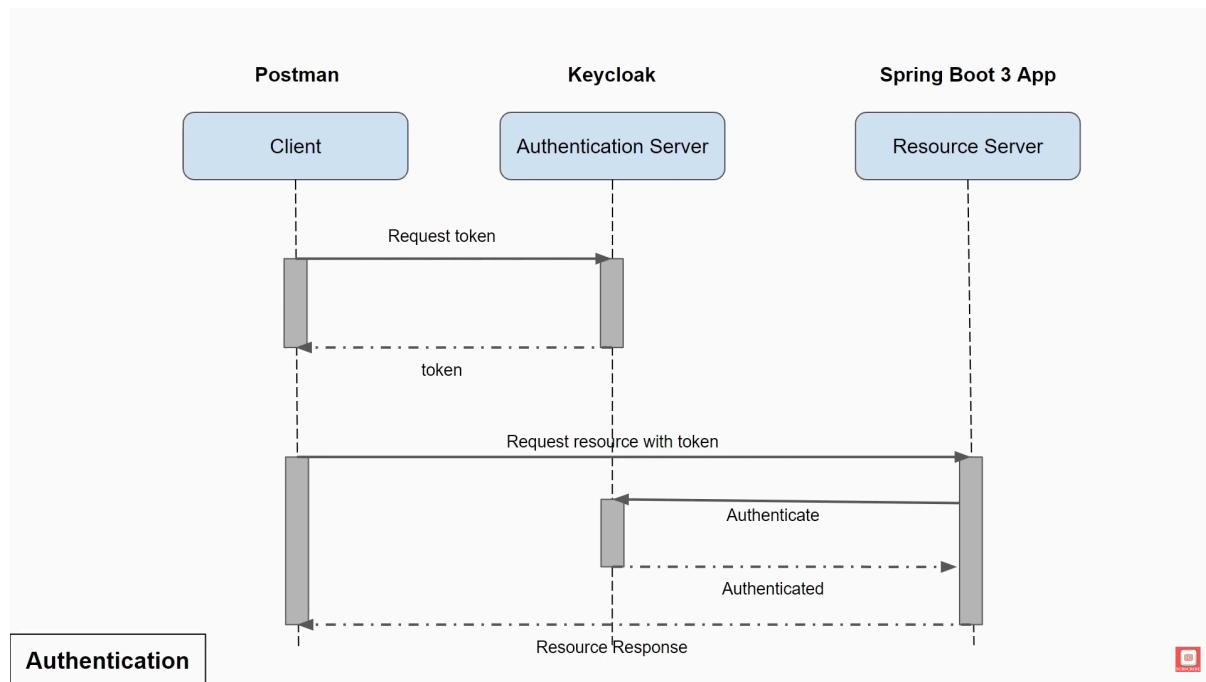
## 2. JWT Tokens:

- JWT is a compact, URL-safe means of representing claims to be transferred between two parties.
- JWTs are encoded as JSON objects and are self-contained, meaning they contain all the necessary information about the user or session within the token itself.
- JWTs typically consist of three parts: a header, a payload (claims), and a signature. The payload contains the claims, such as user ID, expiration time, etc.
- JWTs are often used in stateless authentication systems, where the server does not need to maintain session state. Since JWTs contain all the necessary information, the server can validate and process them without needing to look up additional data.
- JWTs are commonly used in token-based authentication systems, where the client sends the token with each request to authenticate itself.

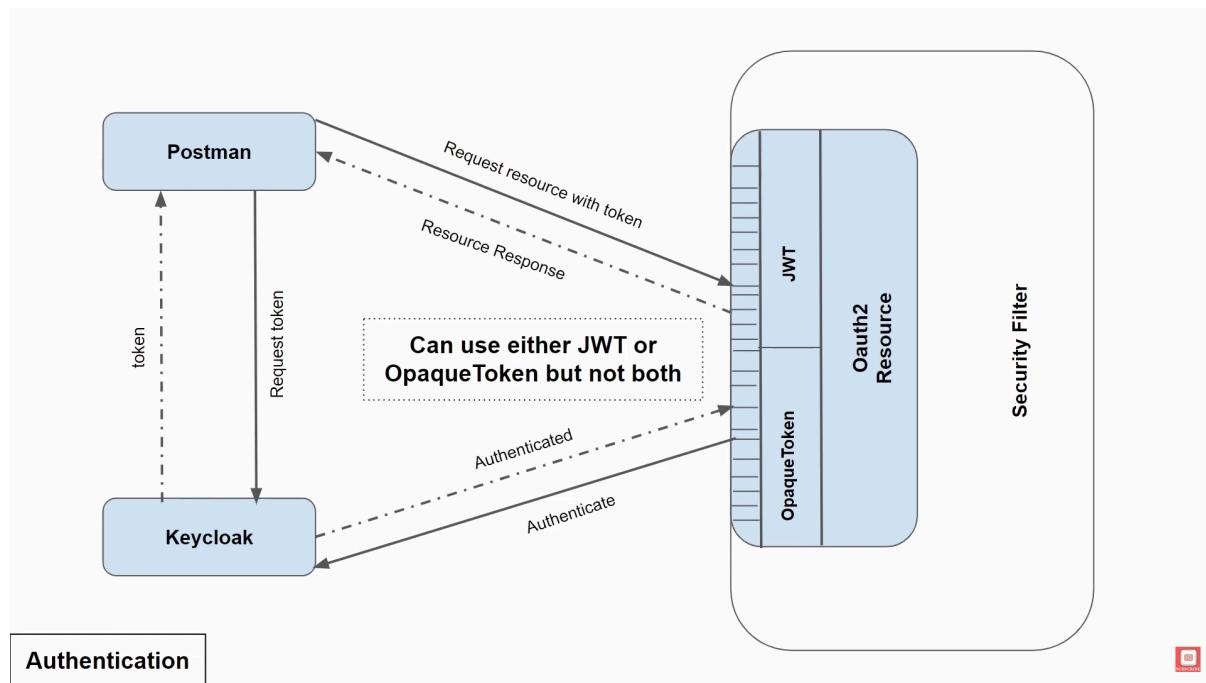
## Create Spring boot Application:



## The Flow of Client to Resource Server.



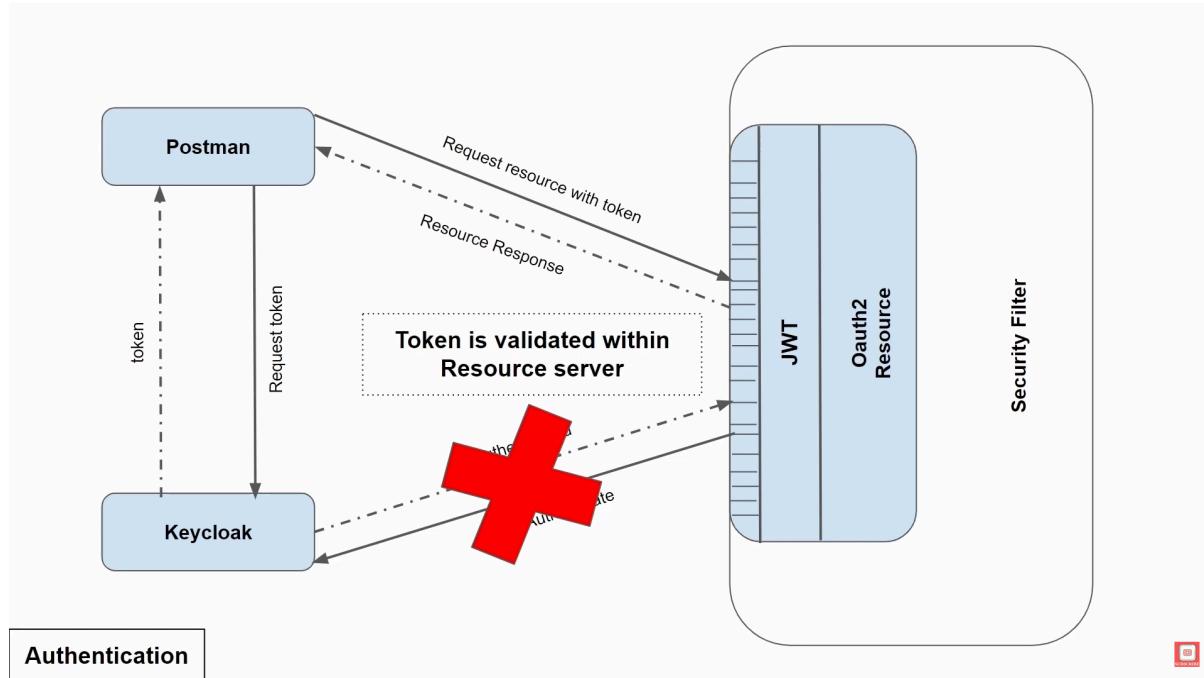
This is another way of Flow for Client to Resource Server.



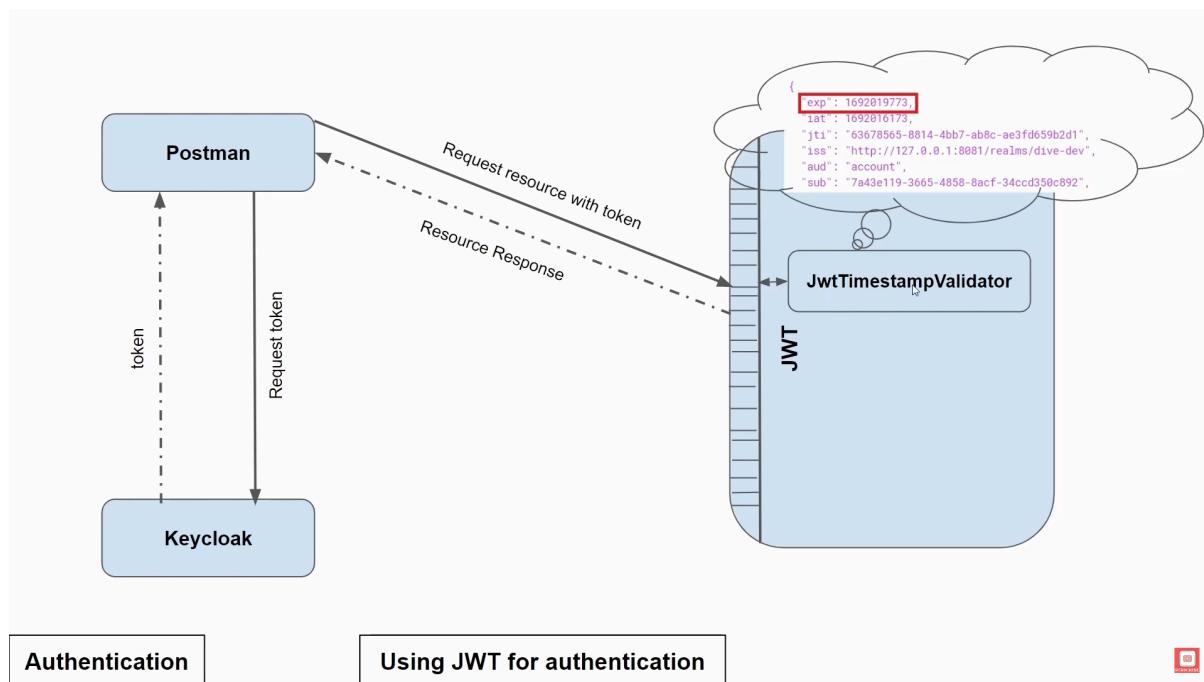
In spring boot Security use Oauth2 Resource and use JWT Token or OPAQUE Token, but in Spring security and Oauth2 either we can use JWT or OPAQUE Token, We can't use Both.

### In JWT Token:

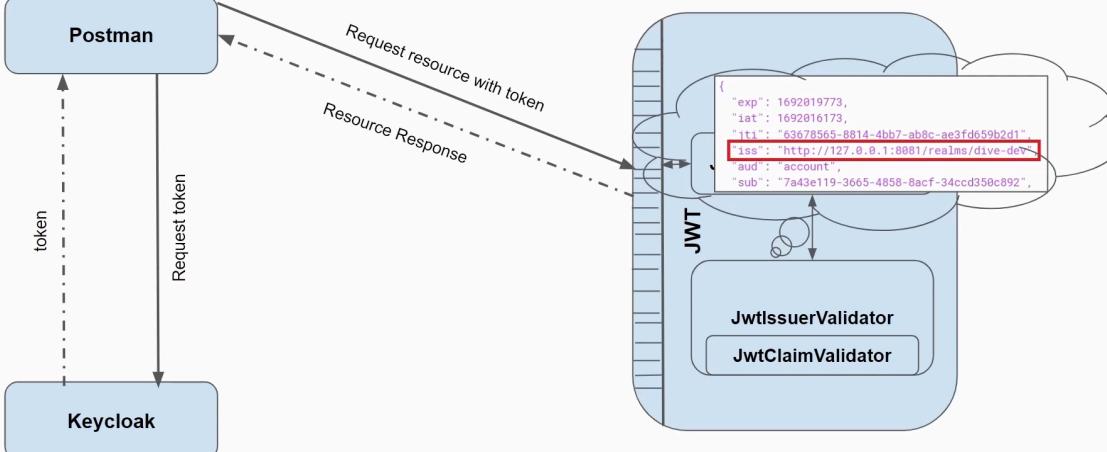
In Spring security and Oauth2 Resource JWT Token is Validated within Resource Server, It will not be sent to the Keycloak Authentication Server.



IN JWTTimestampValidator it used to check the exp date and validate.



JWTIssuerValidator will check the iss value issued by Keycloak.



Authentication

Using JWT for authentication



JwtIssuerValidator - JwtClaimValidator

- Get the value from property `spring.security.oauth2.resource-server.jwt.issuer-uri`.
- Get the value from the access\_token's "iss" property.
- Check if both the strings are equal.
  - If yes then its a valid token.
  - Else the token has invalid issuer.

Using JWT for authentication



But it will not Check the session is Logout for Keycloak or not so the data can be passed. This is a disadvantage in JWT Token.

## Authentication:

In Spring Boot :

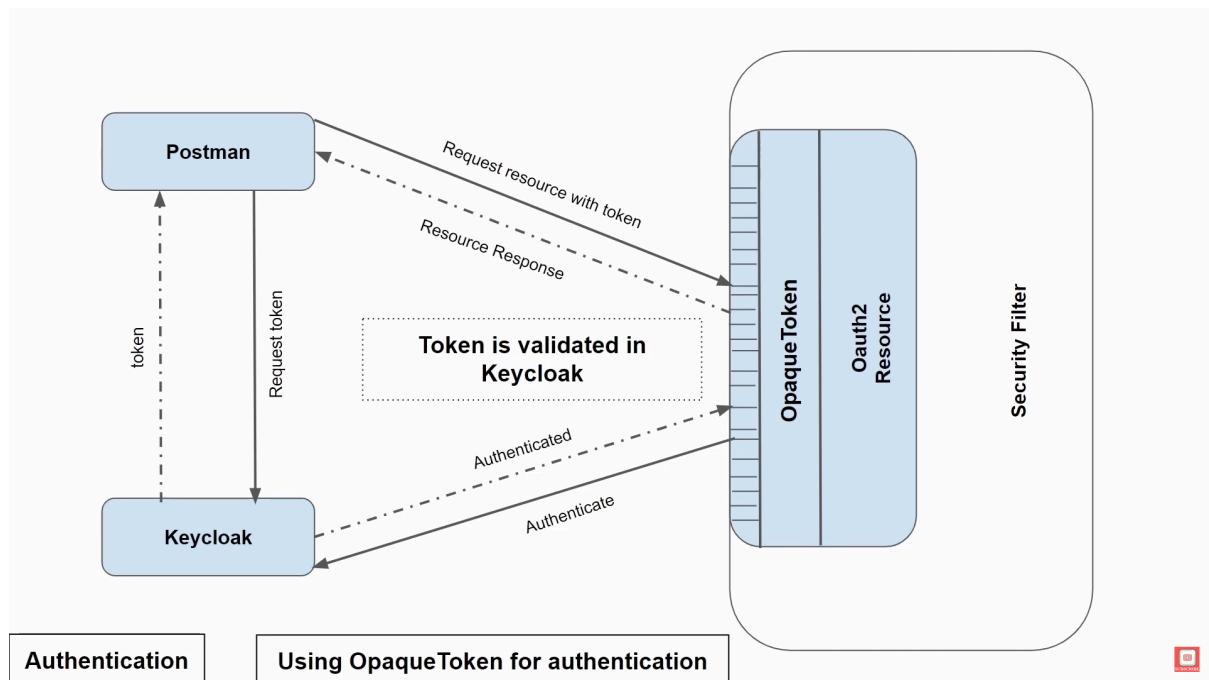
– IN properties file :

```
spring.security.oauth2.resource-server.jwt.issuer-uri=http://localhost:8081/re  
alms/springboot-be
```

– IN security : `.oauth2ResourceServer(oauth2ResourceServerCustomizer ->  
oauth2ResourceServerCustomizer.jwt(Customizer.withDefaults()));`

### OPAQUE TOKEN:

In Spring security and Oauth2 Resource OPAQUE Token is Validated Send's to Authentication Server(Keycloak Server).



From This Three property it gives token validation to Keycloak.

### SpringOpaqueTokenIntrospector

- Get the value from property `spring.security.oauth2.resource-server.opaque-token.introspection-uri` as `introspectURI`.
- Get the value from property `spring.security.oauth2.resource-server.opaque-token.client-id` as `clientId`.
- Get the value from property `spring.security.oauth2.resource-server.opaque-token.client-secret` as `clientSecret`.
- Using RestTemplate object, make an API call using `introspectURI`, `clientId`, and `clientSecret`.

### Using OpaqueToken for authentication



#### In Spring Boot :

– IN properties file :

```
spring.security.oauth2.resource-server.opaque-token.client-id=springboot-dev
spring.security.oauth2.resource-server.opaque-token.client-secret=7P1jVrSko0F9z
PDc0tzEIxRCp9ReyzW7
spring.security.oauth2.resource-server.opaque-token.introspection-uri=http://localhost:8081/realmsspringboot-be/protocol/openid-connect/token/introspect
```

– IN security : `.oauth2ResourceServer(oauth2ResourceServerCustomizer -> oauth2ResourceServerCustomizer.opaqueToken(Customizer.withDefaults()));`

#### Authorization:

In Authorization it will not send to Keycloak Server for JWT and OPAQUE Token.  
Adjusted the JWT Token.

#### Creating New CLAIM in JWT Token in KeyCloak :

## JWT - Method Level Authorization

Incoming JWT token is adjusted to Spring Boot Oauth2 format

```
"realm_access": {  
    "roles": [  
        "default-roles-dive-dev",  
        "manager",  
        "offline_access",  
        "uma_authorization"  
    ]  
},  
"resource_access": {  
    "account": {  
        "roles": [  
            "manage-account",  
            "manage-account-links",  
            "view-profile"  
        ]  
    }  
},  
"scope": "profile email",  
"sid": "ec883715-65ce-4534-ad9e-78e8ecccca88",  
"email_verified": true,  
"name": "Suresh Wadekar",  
"preferred_username": "suresh",
```



## JWT - Method Level Authorization JwtAuthenticationConverter

- Check if token contains property “scope” or “scp” in jwt access token.
  - Checks for “scp” only if “scope” is unavailable.
- Append the prefix “SCOPE\_” to each entry in the value of the property.
  - Works on only the String value having roles separated by space.
  - Or a string array of roles.
- Adds these appended entries as roles in JwtAuthenticationToken class.
- These are then used to authorize each API accordingly.

Authorization



**JWT - Method Level Authorization**

**JwtAuthenticationConverter**

```

"scope" : "profile email"

roles = ["profile" "email"]

Add prefix "SCOPE_"
roles = ["SCOPE_profile" "SCOPE_email"]

```

new JwtAuthenticationToken(jwt, roles)

**Roles do not have "manager"**

**Keycloak - JWT**

```

"realm_access": {
  "roles": [
    "default-roles-dive-dev",
    "manager",
    "offline_access",
    "uma_authorization"
  ]
},
"resource_access": {
  "account": {
    "roles": [
      "manage-account",
      "manage-account-links",
      "view-profile"
    ]
  }
},
"scope": "profile email",
"sid": "ec883715-65ce-4534-ad9e-78e8ecccca88",
"email_verified": true,
"name": "Suresh Wadekar",
"preferred_username": "suresh",

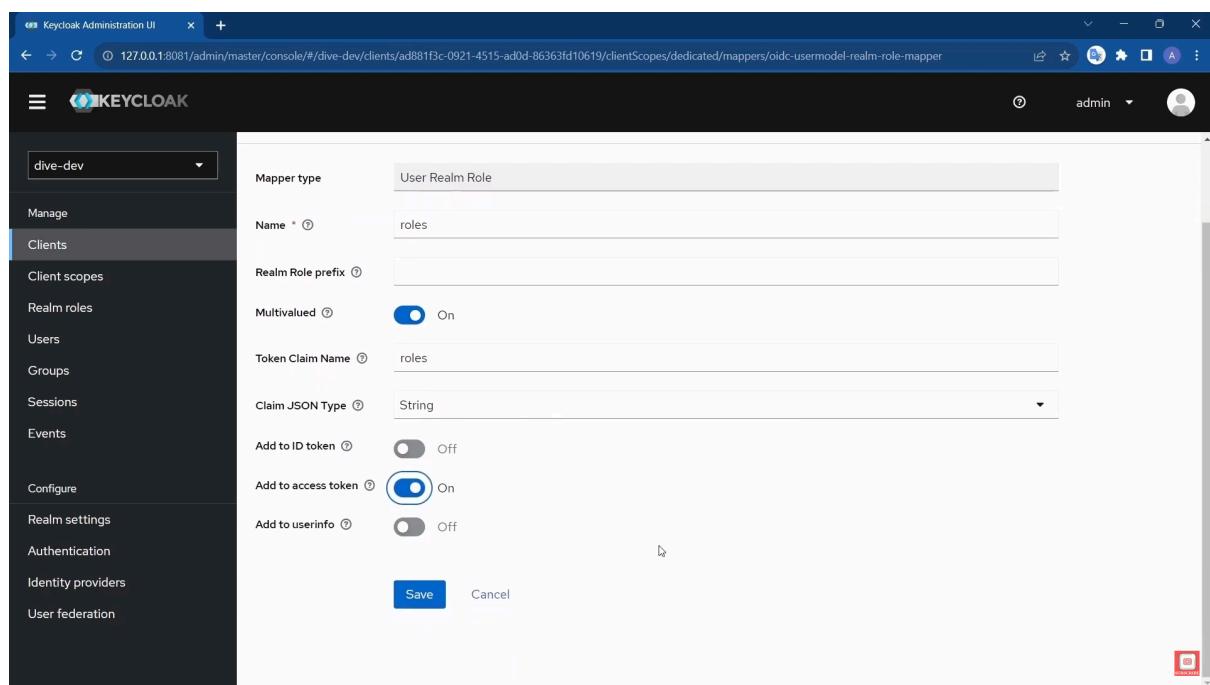
```

**Authorization**

## Creating New Data Inside KeyCloak:

### Adding Role:

Insert Client->client Scope -> default client scope ( [springboot-dev-dedicated](#) ) -> configure new mapper -> user Realm Role



And get roles by getting directly with the JWT Token. And Enable the Swagger in springdoc (springdoc) is directly access by spring boot 3 so we can make less amount of implementation

Ref : <https://springdoc.org/> and <https://reflectoring.io/spring-boot-springdoc/>

## Authentication:

Enable the **Authentication** so It will be enabled.

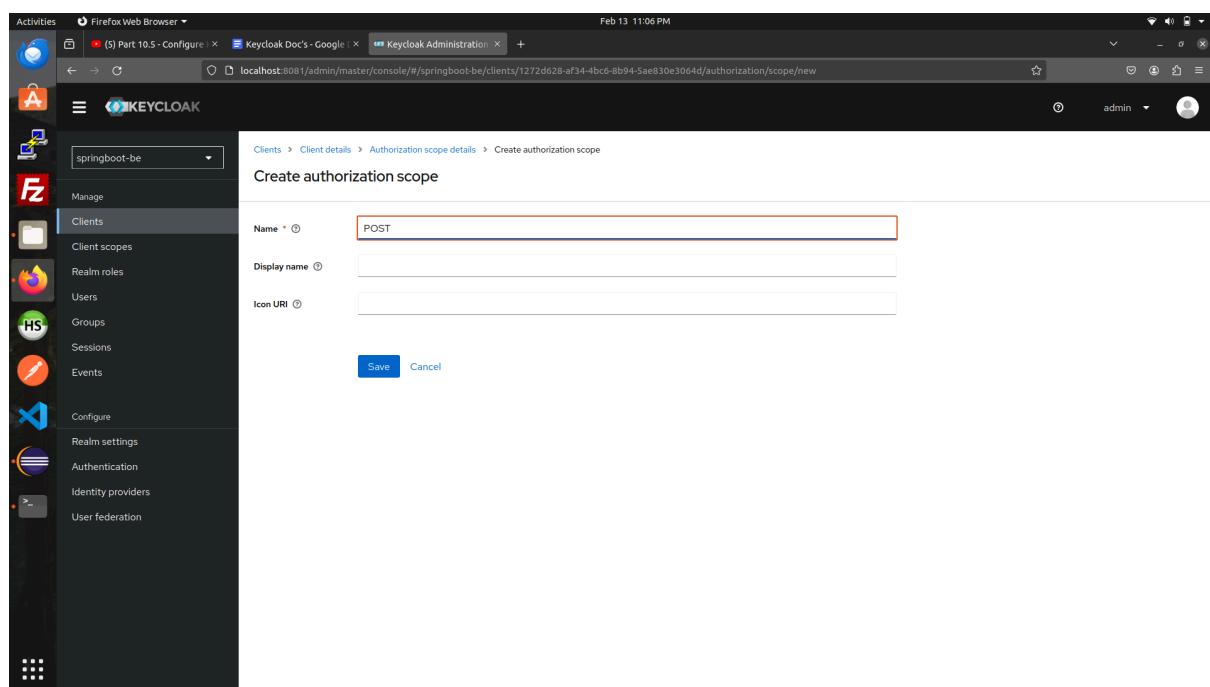
You will see the Authentication Table, go to the Tab and Delete the Resource, Policies if you delete the Policies Permission will be deleted.

Create a new Resource,policies,scope and Permission

## In Scope:

In Scope (we can define POST,GET,PUT methods) we can configure resources.

If we want to use Scope we need to put `"http-method-as-scope":true,` in JSON file in spring boot Application.



## In Resource:

### Resource without Scope :

The screenshot shows the Keycloak Administration interface for creating a new resource. The left sidebar is titled 'Manage' and includes options like Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The 'Clients' option is selected. The main panel is titled 'Create resource' and contains the following fields:

- Owner:** springboot-dev
- Name:** Default policy
- Display name:** (empty)
- Type:** (empty)
- URIs:** /\*
- Authorization scopes:** (empty dropdown)
- Icon URI:** (empty)
- User-Managed access enabled:** Off (radio button)
- Resource attribute:** (instructions: No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair. Add an attribute)

At the bottom are 'Save' and 'Cancel' buttons.

The screenshot shows the Keycloak Administration interface for creating a new resource. The left sidebar is identical to the previous screenshot. The main panel is titled 'myCards Resource' and contains the following fields:

- Owner:** springboot-dev
- Name:** myCards Resource
- Display name:** (empty)
- Type:** (empty)
- URIs:** /myCards
- Authorization scopes:** (empty dropdown)
- Icon URI:** (empty)
- User-Managed access enabled:** Off (radio button)
- Resource attribute:** (instructions: No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair. Add an attribute)

At the bottom are 'Save' and 'Cancel' buttons. A 'Action' dropdown menu is visible at the top right of the main panel.

### Resource with Scope :

Resource will be one i.e API can be one but methods can be different as POST,GET etc.. So,We need to add scope in Authorization scopes.

The screenshot shows the Keycloak Administration interface. On the left, there's a sidebar with various icons and links like Activities, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The 'Clients' link is currently selected. In the main content area, it says 'Clients > Client details > Resource details' and shows a resource named 'myCards Resource'. The 'Owner' is listed as 'springboot-dev'. Under 'Authorization scopes', 'POST' and 'GET' are selected. There's also a note about 'Resource attribute' stating 'No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair.' At the bottom, there are 'Save' and 'Cancel' buttons. A success message 'Resource successfully updated' is displayed at the top right.

## In Policies:

Create a new Policies select Role.

In Add role select (default-roles-springboot-be) because Every user has this.

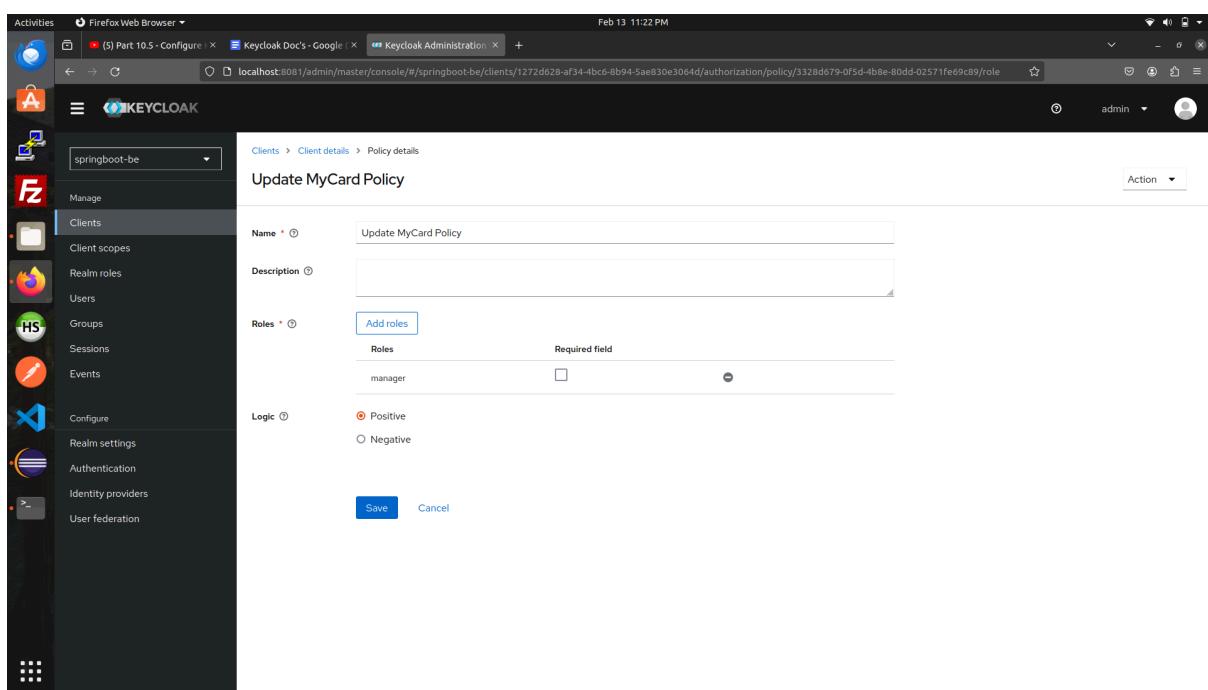
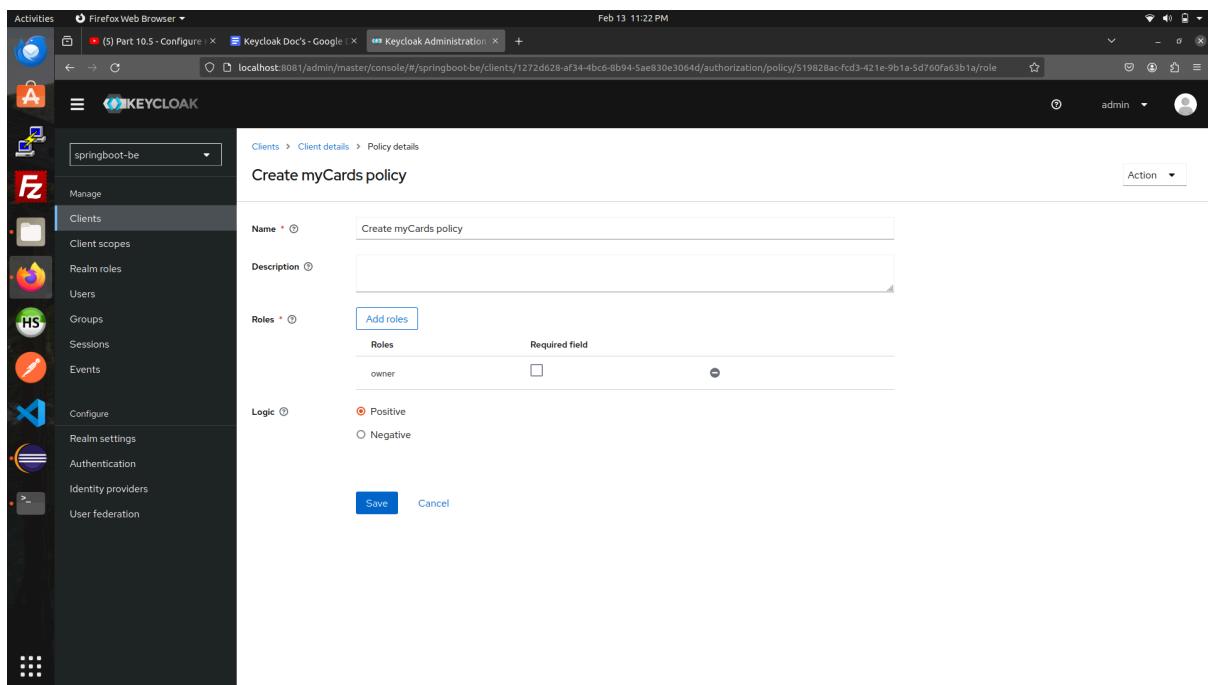
**Policies** without Scope :

The screenshot shows the Keycloak Administration interface in Firefox. The left sidebar is dark-themed and includes icons for Activities, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The 'Clients' icon is highlighted. The main content area shows the 'Create role policy' screen for the 'springboot-be' client. The 'Name' field contains 'Default role policy'. The 'Description' field is empty. Under 'Roles', there is a table with one row: 'default-roles-springboot-be' under 'Roles' and 'Required field' checked. Under 'Logic', the 'Positive' radio button is selected. At the bottom are 'Save' and 'Cancel' buttons.

The screenshot shows the Keycloak Administration interface in Firefox, identical to the previous one but with a different policy name. The left sidebar is dark-themed and includes icons for Activities, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The 'Clients' icon is highlighted. The main content area shows the 'Create role policy' screen for the 'springboot-be' client. The 'Name' field contains 'myCards policy'. The 'Description' field is empty. Under 'Roles', there is a table with one row: 'owner' under 'Roles' and 'Required field' checked. Under 'Logic', the 'Positive' radio button is selected. At the bottom are 'Save' and 'Cancel' buttons. A 'Policy details' link is visible above the save buttons.

## Policies with Scope :

We have to create different policies like manager, owner ect...

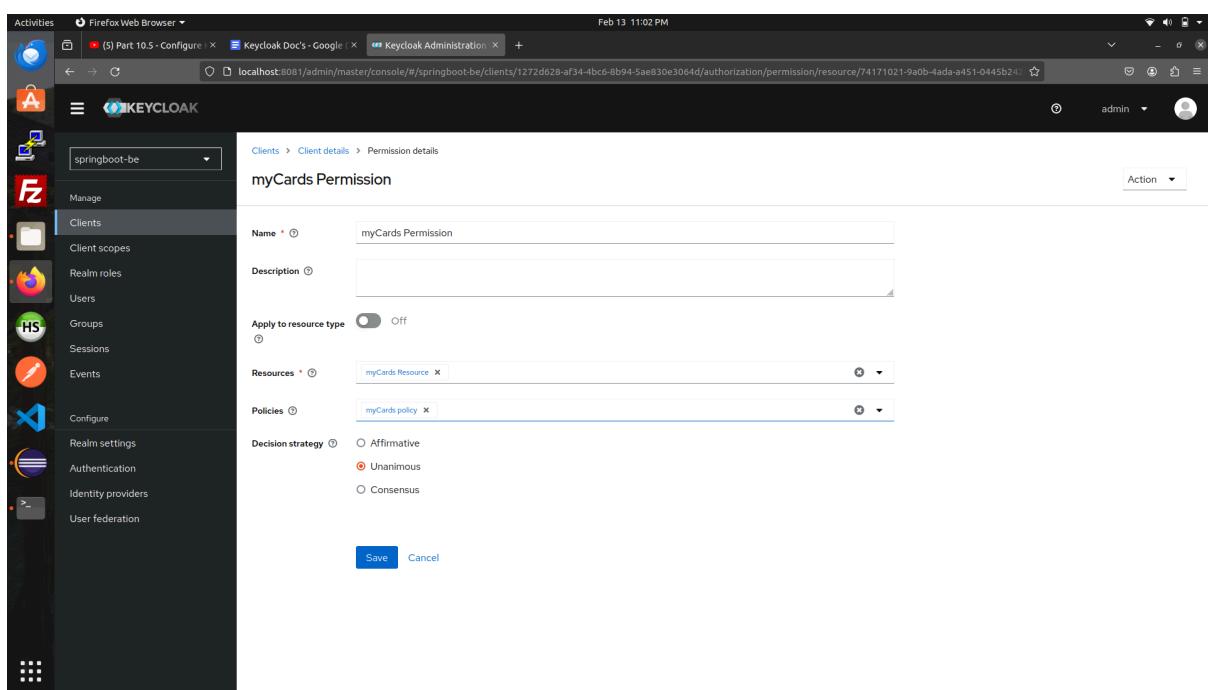
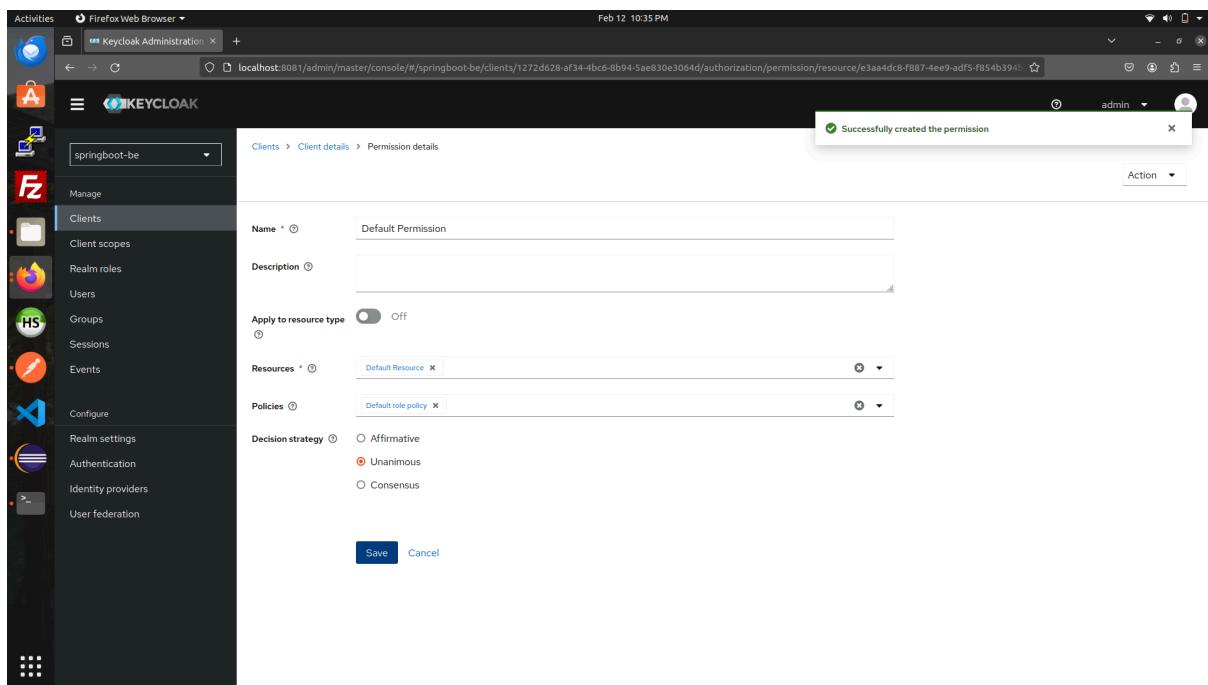


## In Permission:

Two types of Permission are Resource-base Permission and Scope-base Permission.

We created Resource only Resource-base Permission is enabled.

## Permission without Scope :



## Permission with Scope :

We have to create different **Permission** like POST, GET etc...  
Create permission select Add bases cope.

Activities Firefox Web Browser ▾

localhost:8081/admin/master/console/#/springboot-be/clients/1272d628-af34-4bc6-8b94-5ae830e3064d/authorization/policy/3328d679-0f5d-4b8e-80dd-02571fe69c89/role

Keycloak Administration

Client details > Policy details

Update MyCard Policy

Name \* Update My Card Policy

Description

Roles \* Add roles

manager Required field

Logic \* Positive

Save Cancel

Clients

Manage

Client scopes

Realm roles

Users

Groups

Sessions

Events

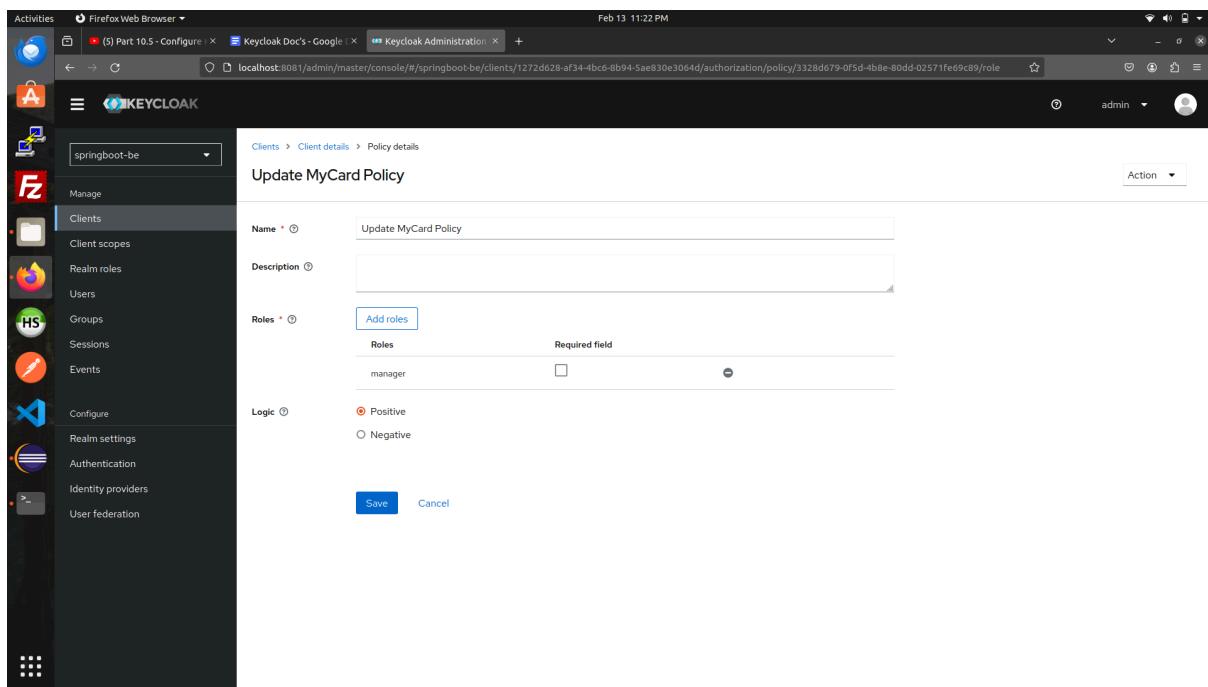
Configure

Realm settings

Authentication

Identity providers

User federation



Activities Firefox Web Browser ▾

localhost:8081/admin/master/console/#/springboot-be/clients/1272d628-af34-4bc6-8b94-5ae830e3064d/authorization/permission/new/scope

Keycloak Administration

Clients > Client details > Create permission

Create scope-based permission

Name \* Create myCards permission

Description

Apply to resource type Off

Resources myCards Resource

Authorization scopes POST

Policies Create myCards policy

Decision strategy \* Unanimous

Save Cancel

Clients

Manage

Client scopes

Realm roles

Users

Groups

Sessions

Events

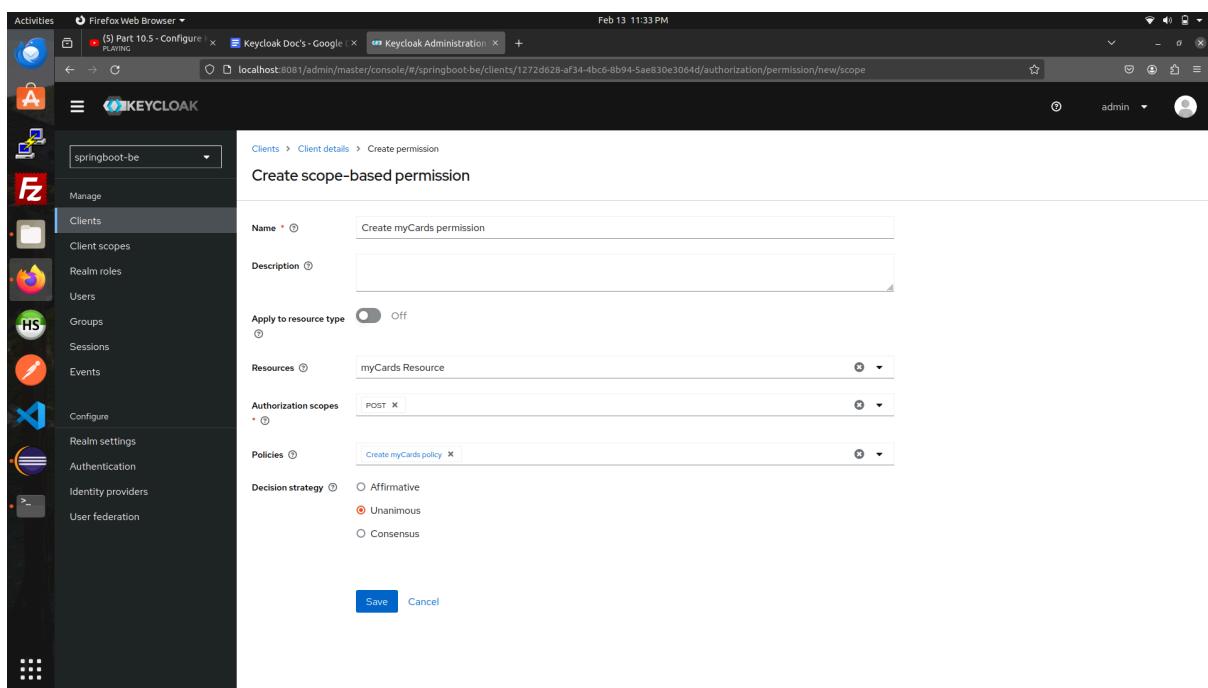
Configure

Realm settings

Authentication

Identity providers

User federation



## In Authentication:

### Different Type of Policies:

Role -> Role base **Authentication**

User -> User base **Authentication**

Group -> If we have lot of user for one API

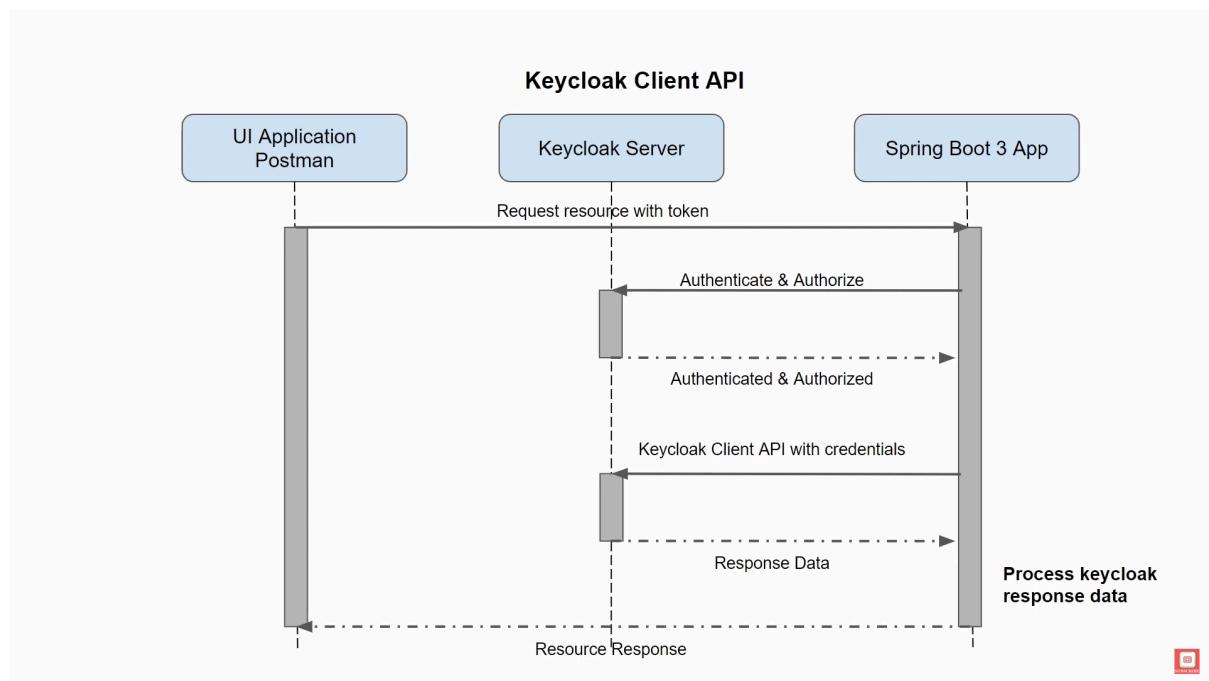
eg: /mycard has to access different type of user need to access this API we can create

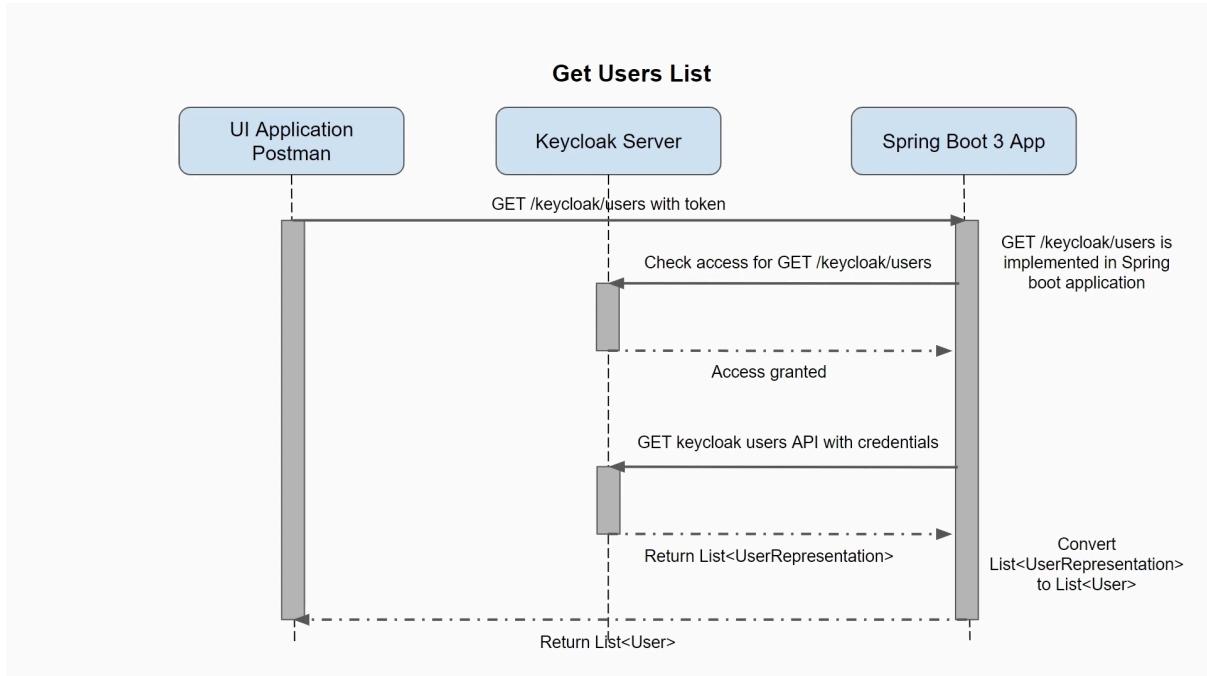
Group and add the user and we can also add child group so the all user can access this API

Time -> we can set time for API so that time only API can access. And it work

Regex -> It works on regex patterns,for details see YOUTUBE videos.

## In springBoot:





### Create Admin as User:

Need to create admin and password and role as ADMIN

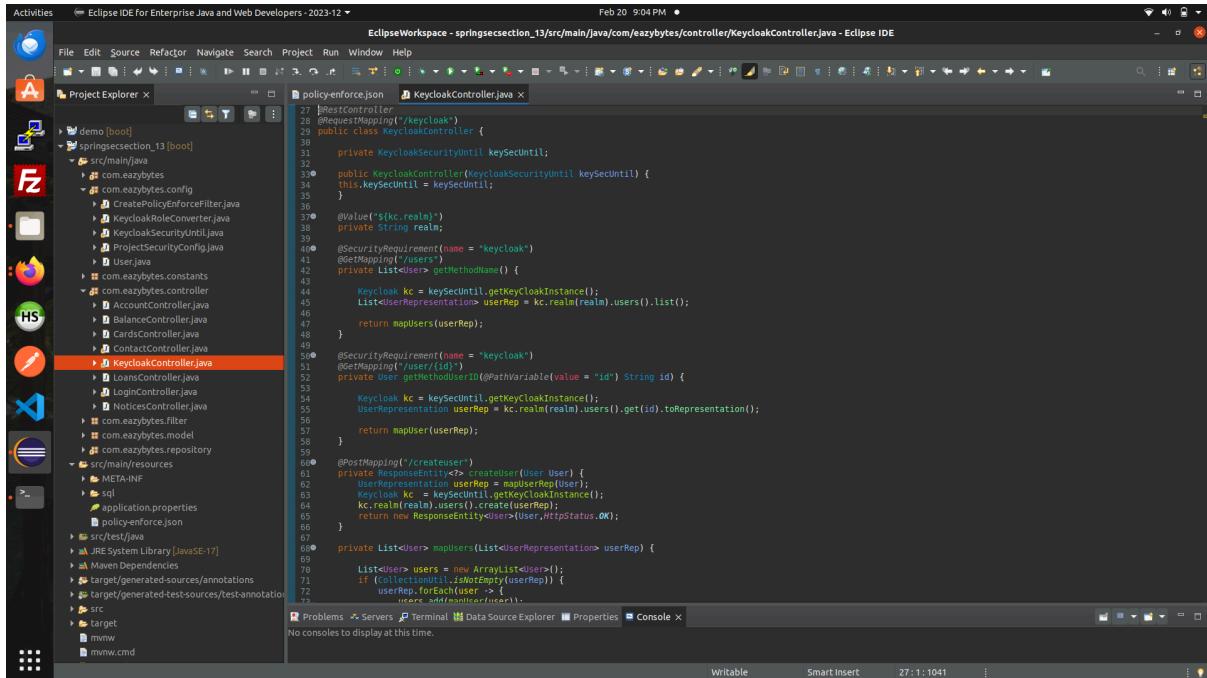
In Role mapping Click Assign Role -> Filter by realm role -> Filter by Client -> search for realm-admin.

Add Role as ADMIN.

The screenshot shows the Keycloak Administration interface in a Firefox browser. The URL is `localhost:8081/admin/master/console/#/springboot-be/users/ad38b498-9bd1-4ceb-86c9-43a214b51b11/role-mapping`. The left sidebar is collapsed. The main area shows the 'User details' for a user named 'Admin'. The 'Role mapping' tab is selected. The table lists roles assigned to the user:

Name	Inherited	Description
admin	False	-
default-roles-springboot-be	False	<code>#{role_default-roles}</code>
realm-management	realm-admin	<code>#{role_realm-admin}</code>

## Need to create Resource,Scope,Policy and Permission for /keycloak API's

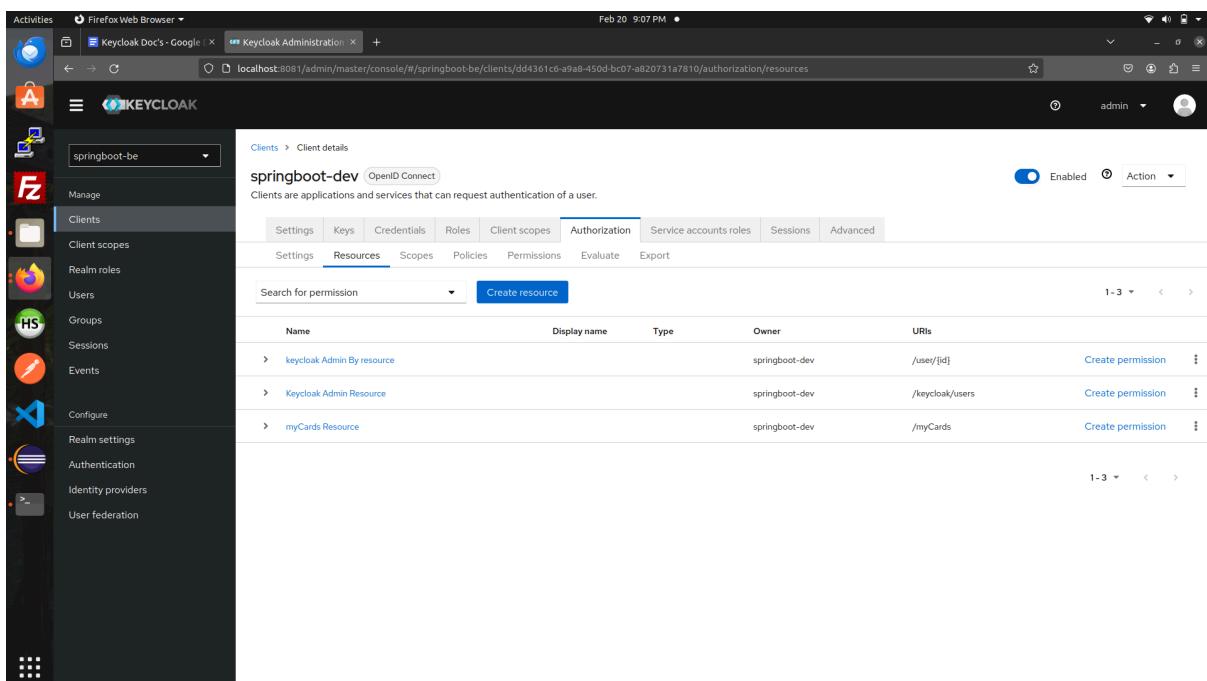


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Eclipse IDE for Enterprise Java and Web Developers - 2023-12
- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Project Explorer:** Shows the project structure for "springsecsection\_13". The "KeycloakController.java" file is selected.
- Code Editor:** Displays the Java code for the "KeycloakController" class, which interacts with Keycloak via a "KeycloakSecurityUntil" instance.
- Bottom Status Bar:** Writable, SmartInsert, 27:1:1041

```
Activities Eclipse IDE for Enterprise Java and Web Developers - 2023-12 • EclipseWorkspace - springsecsection_13/src/main/java/com/eazybytes/controller/KeycloakController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer × policy-enforce.json KeycloakController.java
demo [boot]
src/main/java
  com.eazybytes
    config
      CreatePolicyForceFilter.java
      KeycloakRoleConverter.java
      KeycloakSecurityUntil.java
      ProjectSecurityConfig.java
    User.java
    constants
    controller
      AccountController.java
      BalanceController.java
      CardsController.java
      ContactController.java
      KeycloakController.java
      LoansController.java
      LoginController.java
      NoticesController.java
    filter
    model
    repository
src/main/resources
  META-INF
  sql
    application.properties
    policy-enforce.json
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
src
target
  mvnw
  mvnw.cmd
policeforce.json
  demo [boot]
  src/main/java
    com.eazybytes
      config
        CreatePolicyForceFilter.java
        KeycloakRoleConverter.java
        KeycloakSecurityUntil.java
        ProjectSecurityConfig.java
      User.java
      constants
      controller
        AccountController.java
        BalanceController.java
        CardsController.java
        ContactController.java
        KeycloakController.java
        LoansController.java
        LoginController.java
        NoticesController.java
      filter
      model
      repository
    src/main/resources
      META-INF
      sql
        application.properties
        policy-enforce.json
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  src
  target
    mvnw
    mvnw.cmd
  Problems Servers Terminal Data Explorer Properties Console ×
No consoles to display at this time.
Writable SmartInsert 27:1:1041
```

### Create Resource for /keycloak API:



The screenshot shows the Keycloak Admin UI interface with the following details:

- Header:** Firefox Web Browser, Keycloak Doc's - Google, Keycloak Administration, localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/resources, admin
- Sidebar:** Clients (selected), Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, User federation.
- Client Details:** Client name is "springboot-dev" (OpenID Connect).
- Authorization Tab:** Enabled.
- Resources Sub-Tab:** Selected. Shows a table of existing resources:

Name	Display name	Type	Owner	URIs	Action
keycloak Admin By resource			springboot-dev	/user/{id}	Create permission
Keycloak Admin Resource			springboot-dev	/keycloak/users	Create permission
myCards Resource			springboot-dev	/myCards	Create permission

Activities Firefox Web Browser Keycloak Administration type of models in spring + localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/resource/406c38ac-9698-4bbc-8ec1-eca6dacad326 admin Action

springboot-be Manage Clients Client scopes Realm roles Users Groups Sessions Events Configure Realm settings Authentication Identity providers User federation

### Keycloak Admin Resource

Owner: springboot-dev

Name: Keycloak Admin Resource

Type:

URIs: /keycloak/users

Authorization scopes: GET, POST

Icon URI:

User-Managed access enabled: Off

Resource attribute: No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair.

Add an attribute

Save Cancel

This screenshot shows the 'Resource details' configuration for the 'Keycloak Admin Resource'. The 'Name' field is set to 'Keycloak Admin Resource'. The 'URIs' field contains '/keycloak/users'. The 'Authorization scopes' dropdown includes 'GET' and 'POST'. A note at the bottom indicates no attributes have been defined yet, with a link to 'Add an attribute'.

Activities Firefox Web Browser Keycloak Administration type of models in spring + localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/resource/8892cbc1-e1a1-4283-afe4-79ad2fd3fdd admin Action

springboot-be Manage Clients Client scopes Realm roles Users Groups Sessions Events Configure Realm settings Authentication Identity providers User federation

### keycloak Admin By resource

Owner: springboot-dev

Name: keycloak Admin By resource

Type:

URIs: /user/{id}

Authorization scopes:

Icon URI:

User-Managed access enabled: Off

Resource attribute: No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair.

Add an attribute

Save Cancel

This screenshot shows the 'Resource details' configuration for the 'keycloak Admin By resource'. The 'Name' field is set to 'keycloak Admin By resource'. The 'URIs' field contains '/user/{id}'. The 'Authorization scopes' dropdown is empty. A note at the bottom indicates no attributes have been defined yet, with a link to 'Add an attribute'.

**Create resource**

**Owner:** springboot-dev

**Name:** Keycloak Main Resource

**Display name:**

**Type:**

**URIs:** /keycloak/ [ Add URI ]

**Authorization scopes:** POST, DELETE, GET, Show

**User-Managed access enabled:** Off

**Resource attribute:** No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair. [ Add an attribute ]

**Save** **Cancel**

Create Scope for /keycloak API:

**Create Scope as POST,GET,PUT (or) UPDATE and DELETE.**

**springboot-dev** [ OpenID Connect ]

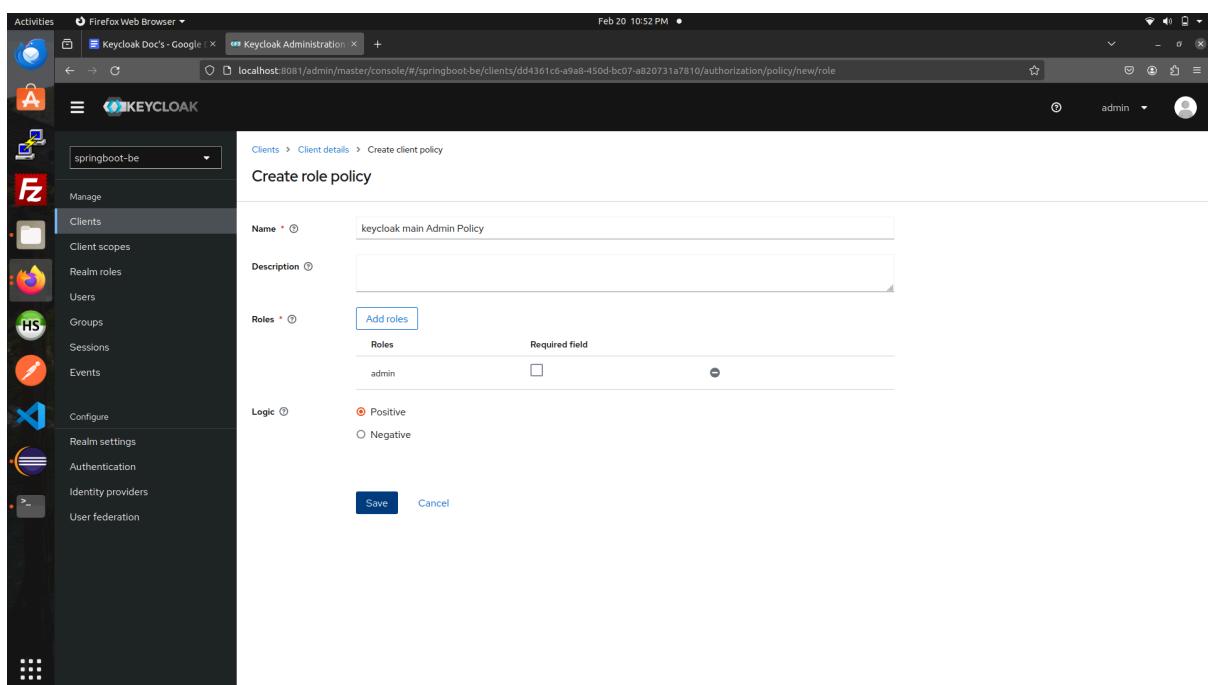
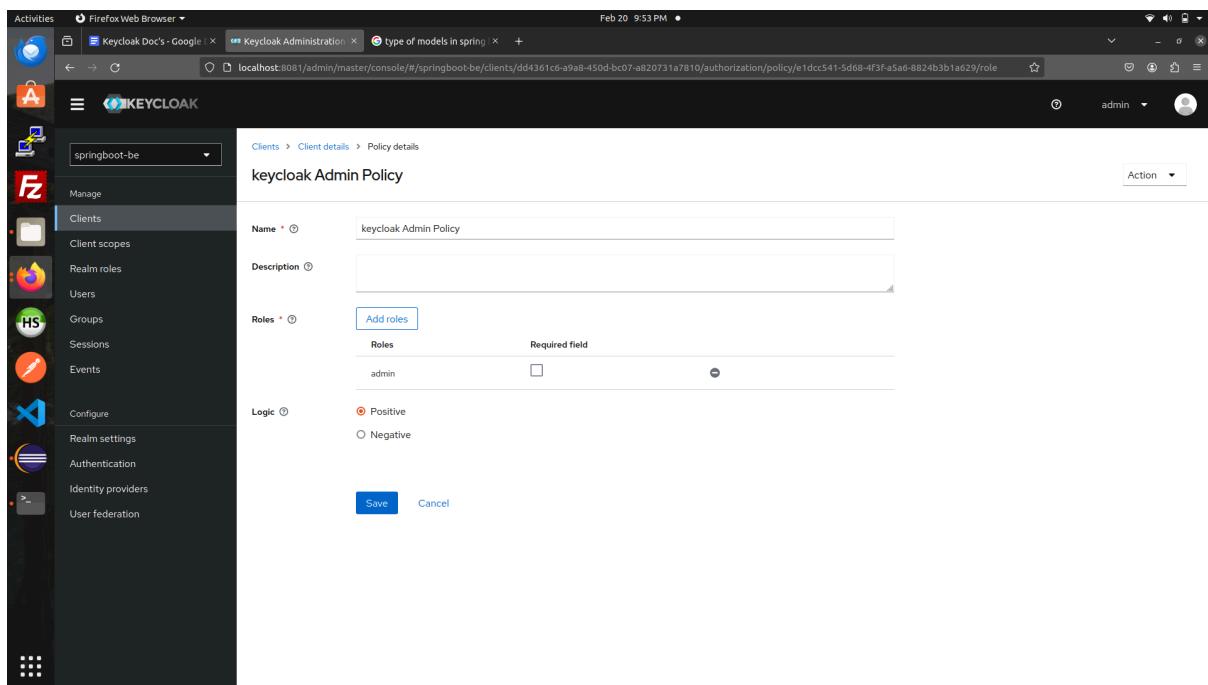
Clients are applications and services that can request authentication of a user.

**Authorization** [ Enabled ] **Action**

**Scopes**

Name	Display name	Actions
DELETE		[ Create permission ]
GET		[ Create permission ]
POST		[ Create permission ]
UPDATE		[ Create permission ]

Create Policy For /keycloak API:



Create Permission for /keycloak API:  
While creating permission click Resource Base.

The screenshot shows the Keycloak Administration interface in a Firefox browser. The URL is `localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/permissions`. The left sidebar is dark-themed and shows the following navigation items:

- Activities
- Manage
- Clients (selected)
- Client scopes
- Realm roles
- Users
- Groups
- Sessions
- Events
- Configure
- Realm settings
- Authentication
- Identity providers
- User federation

The main content area has a light background. It displays the "Client details" for "springboot-dev" (OpenID Connect). The "Authorization" tab is selected, showing the "Permissions" sub-tab. A table lists four permissions:

Name	Type	Associated policy	Description
> create myCards Permission	Scope-Based	Create myCards policy	
> keycloak Admin Permission	Resource-Based	keycloak Admin Policy	
> keycloak By Admin Permission	Resource-Based	keycloak By Admin Policy	
> View myCards Permission	Scope-Based	view myCards policy	

The screenshot shows the Keycloak Administration interface in a Firefox browser. The URL is `localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/permission/resource/4b3db8a2-3f42-42d0-99c5-f9209e9f.../role/e4061e52-d7dd-484a-9e2e-1ec881db00c8`. The left sidebar is dark-themed and shows the same navigation items as the previous screenshot.

The main content area shows the "Permission details" for "keycloak By Admin Permission". The form fields are:

- Name: keycloak By Admin Permission
- Description: (empty)
- Apply to resource type: Off
- Resources: keycloak By Admin Policy
- Policies: keycloak By Admin Po... (with a dropdown arrow)
- Decision strategy:
  - Affirmative
  - Unanimous** (selected)
  - Consensus

At the bottom are "Save" and "Cancel" buttons.

The screenshot shows the Keycloak Administration interface in a Firefox browser. The URL is `localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/permission/new/resource`. The left sidebar is dark-themed and includes icons for Activities, Keycloak Docs - Google, Keycloak Administration, Manage, Clients (selected), Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The main content area is titled "Create resource-based permission". It has fields for Name (keycloak Main Admin Permission), Description, Apply to resource type (Off), Resources (Keycloak Main Resource...), Policies (keycloak main Admin ...), Decision strategy (Unanimous selected), and Save/Cancel buttons.

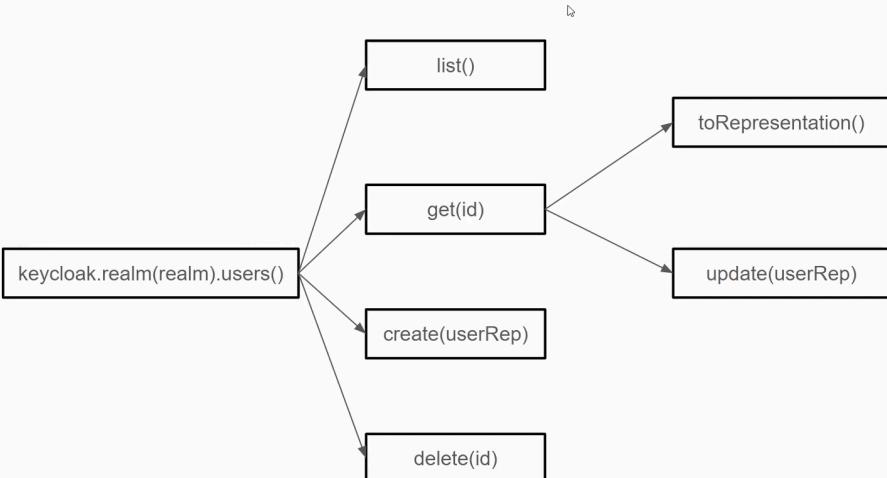
## CRUD Operation Flow of Keycloak:

### USER:

The diagram illustrates the User CRUD operation flow in Keycloak. It starts with a red arrow pointing from the "User" section title to the "Users" page. On the left, a sidebar shows a navigation tree: "dive-dev" (selected), Manage, Clients, Client scopes, Realm roles, **Users** (selected), Groups, Sessions, Events. A red box highlights the "Users" item in the sidebar. A red arrow points from the "Users" item to the "User list" on the right. The "User list" page displays a table of users:

	Username	Email	Last name
<input type="checkbox"/>	admin	admin@admin.com	-
<input type="checkbox"/>	akbar	akbar@test.com	Alahbadi
<input type="checkbox"/>	amar	amar@demo.com	Verma
<input type="checkbox"/>	anthony	anthony@test.com	Gunsalvis

## User CRUD (UserRepresentation)



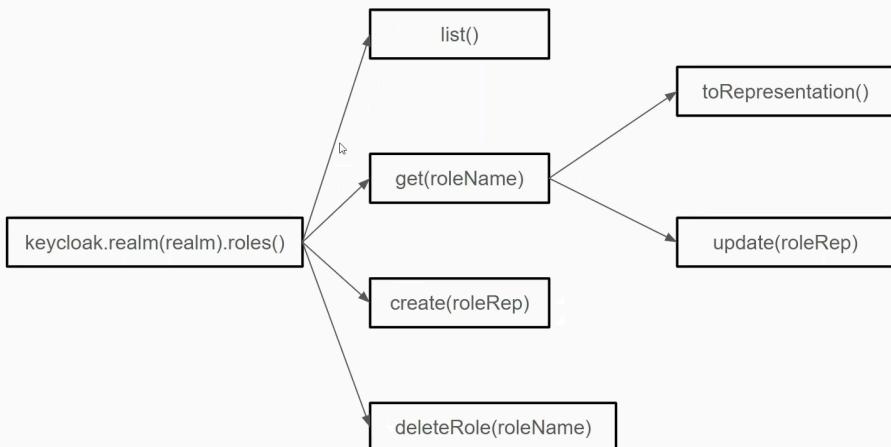
## ROLES:

### Role

The screenshot shows the Keycloak administration interface. The left sidebar is visible with items like Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, and Realm settings. The 'Realm roles' item is highlighted with a red box and has a red arrow pointing to it from the main content area. The main content area is titled 'Realm roles' and contains a table of roles defined for the current realm.

Role name	Composite
admin	False
default-roles-dive-dev	True
HELLO	False
manager	False
offline_access	False
owner	False
uma_authorization	False

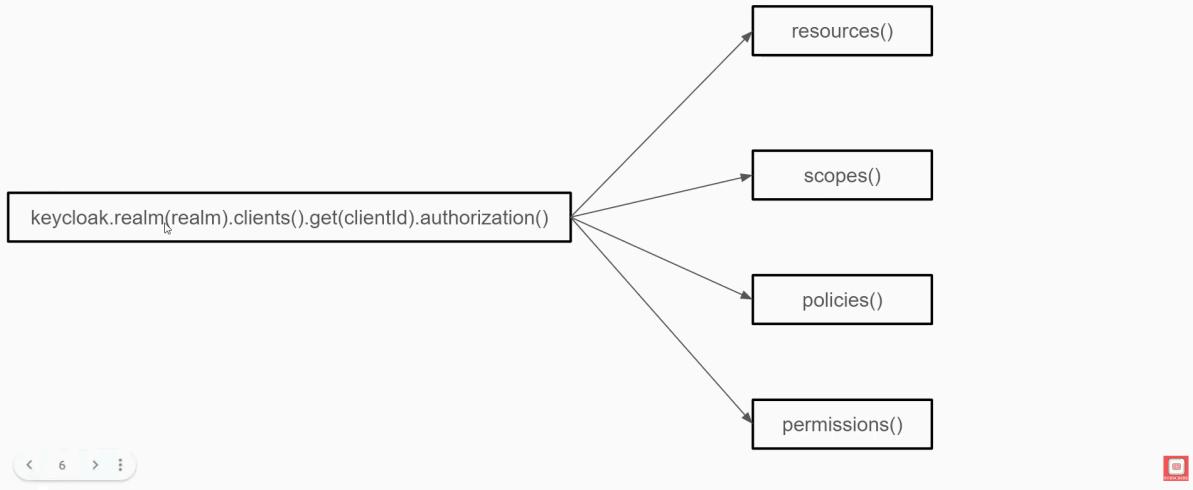
## Role CRUD (RoleRepresentation)



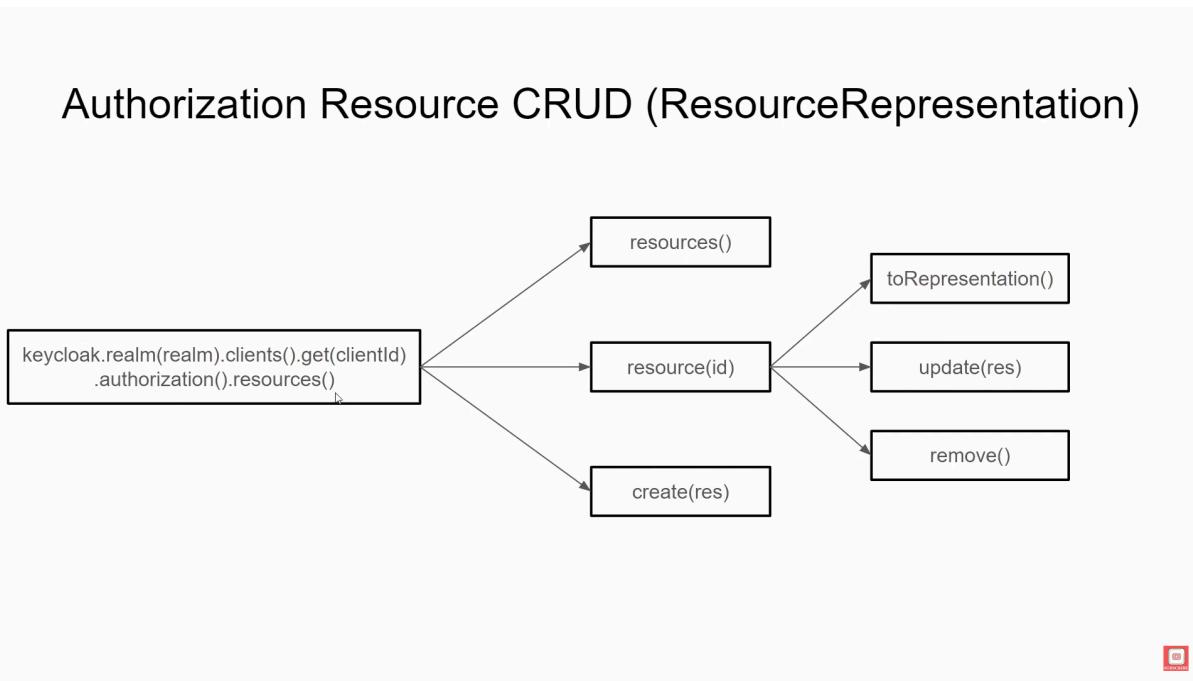
## AUTHENTICATION:

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu is visible with items like `dive-dev`, `Clients`, `Client scopes`, `Realm roles`, `Users`, `Groups`, `Sessions`, and `Events`. The `Clients` item is highlighted with a red box. In the main content area, the title is `Authentication`. The URL in the browser is `Clients > Client details`. The client name is `springboot-be` (with `OpenID Connect` badge). Below the client name, a note says: `Clients are applications and services that can request authentication of a user.` The tab bar at the top has several tabs: `Settings`, `Keys`, `Credentials`, `Roles`, `Client scopes`, `Authorization` (which is highlighted with a red box), and `Service accounts`. Below the tabs, there are buttons for `Import` and `Import`. Under the `Authorization` tab, there are sections for `Policy enforcement mode` (radio buttons for `Enforcing`, `Permissive`, and `Disabled`) and `Resources`, `Scopes`, `Policies`, and `Permissions` (all of which are highlighted with red boxes).

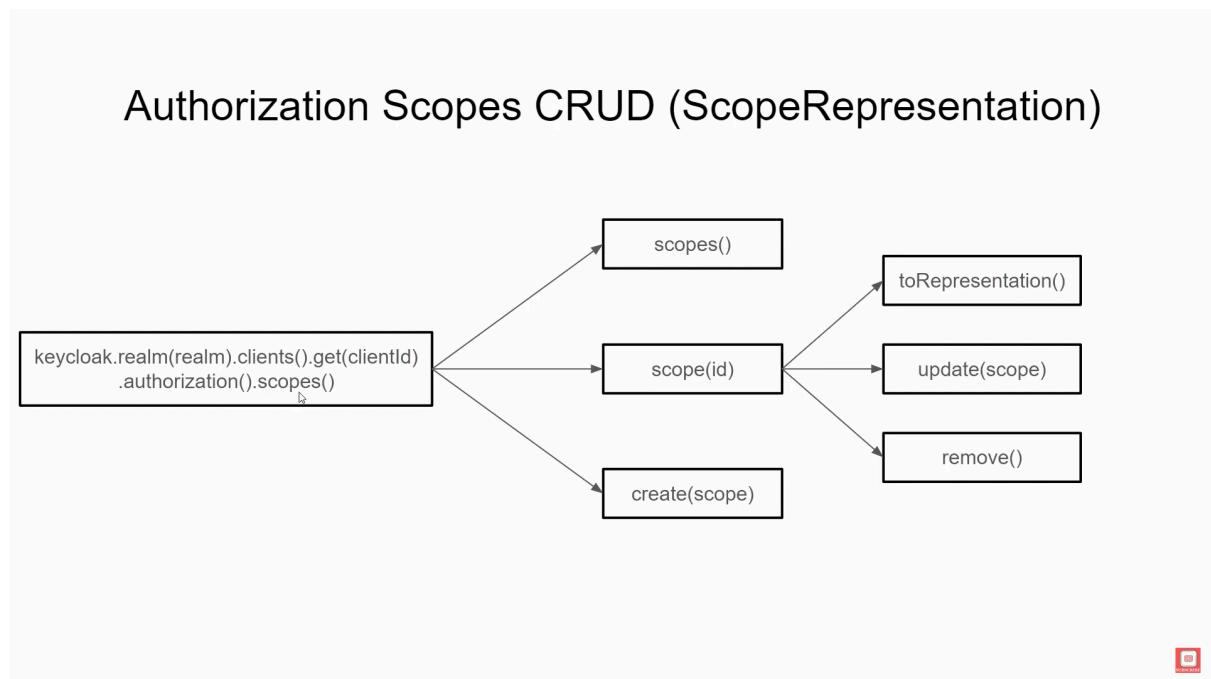
## Authorization



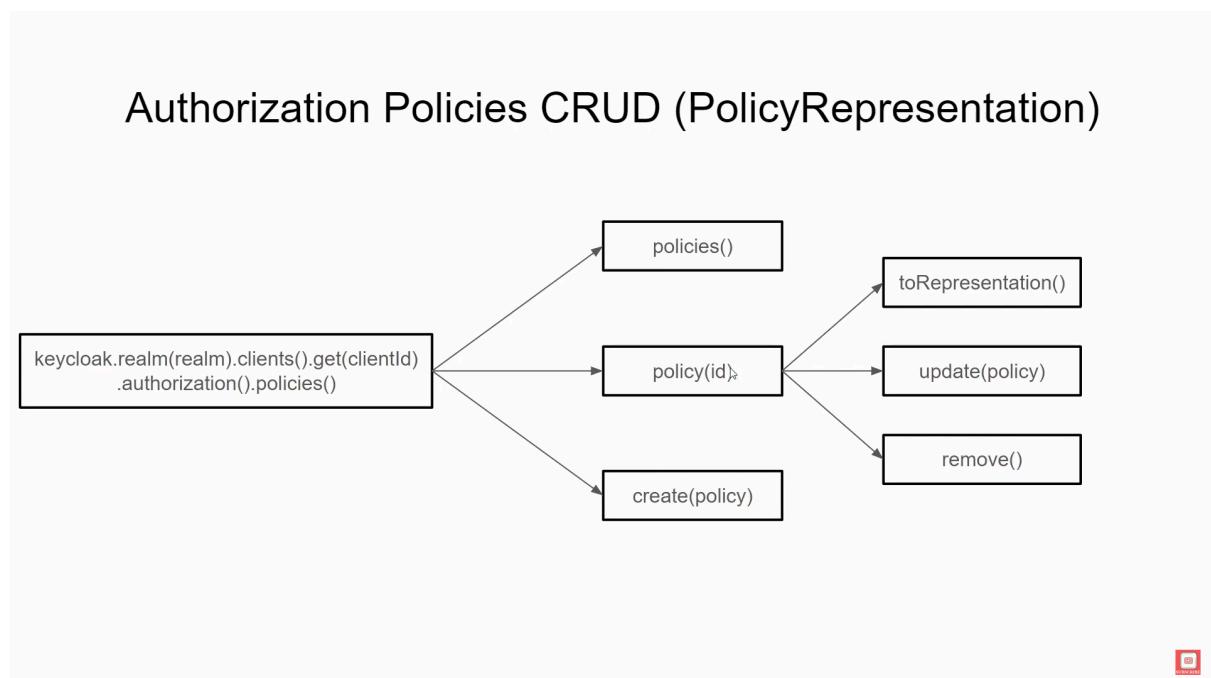
## RESOURCES :



## SCOPE:



## POLICIES:

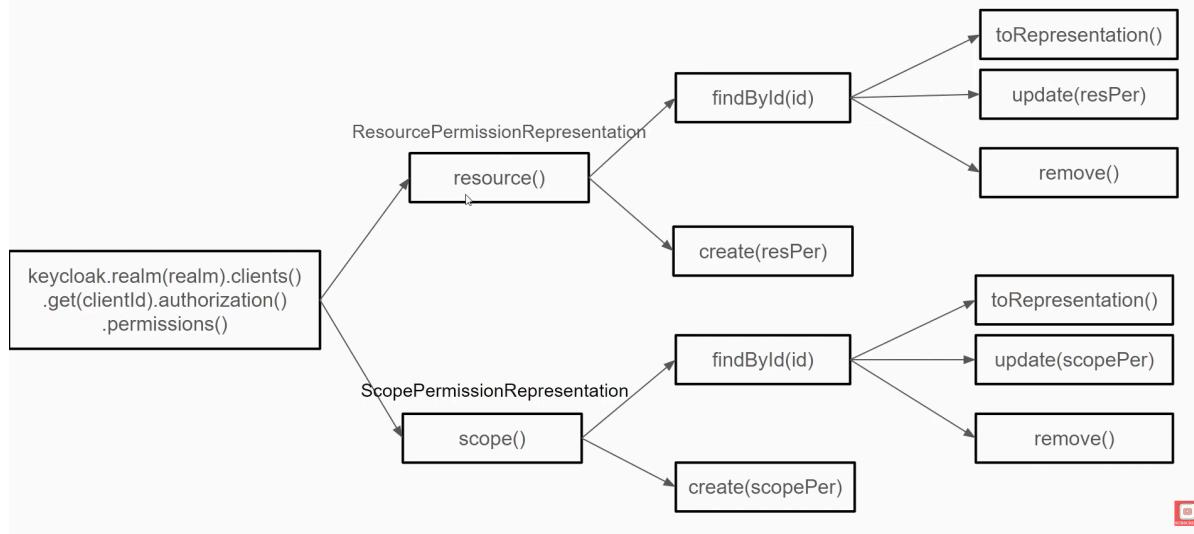


## PERMISSION:

## Authorization Permission

The screenshot shows the Keycloak Authorization Permissions management interface. At the top, there are tabs: Settings, Keys, Credentials, Roles, Client scopes, Authorization (which is highlighted with a red box), Service accounts roles, and Sessions. Below these are sub-tabs: Settings, Resources, Scopes, Policies, Permissions (also highlighted with a red box), Evaluate, and Export. A search bar labeled "Search for permission" is present. A prominent blue button labeled "Create permission" is at the top right. Two options are highlighted with red boxes: "Create resource-based permission" and "Create scope-based permission". The main table lists various permissions with columns for Name, Associated policy, and Type (Scope-Based, Resource-Based). Examples include "Create Menu Permission" (Associated with "Create Menu Policy"), "Create Restaurant Permission" (Associated with "Create Restaurant Policy"), and "Keycloak Admin Permission" (Associated with "Keycloak Admin Policy").

## Authorization Permission CRUD



## **SSO (SINGLE SIGN ON):**

### **Features**

→ Login in one App. and use the token for other apps

→ Use third party login instead of the App.

Two types of sso is :

Login in one App Eg: If you have more than one product and Login for all users is the same, If you login to One of the products they can access for other products.

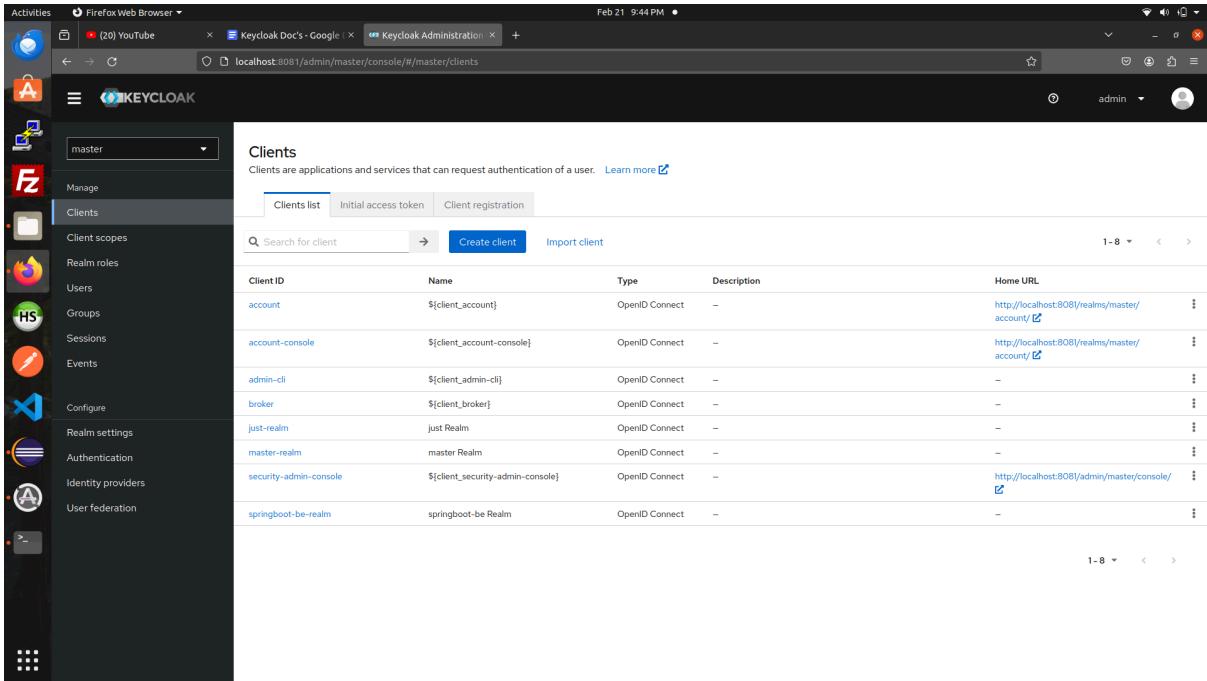
Use Third Party Login Eg : user can login by using Third party App (Github,Google,fb etc..) (you can visit the YOUTUBE : Dive into development)

**The Realm should be the same but Different clients need to be for sso.**

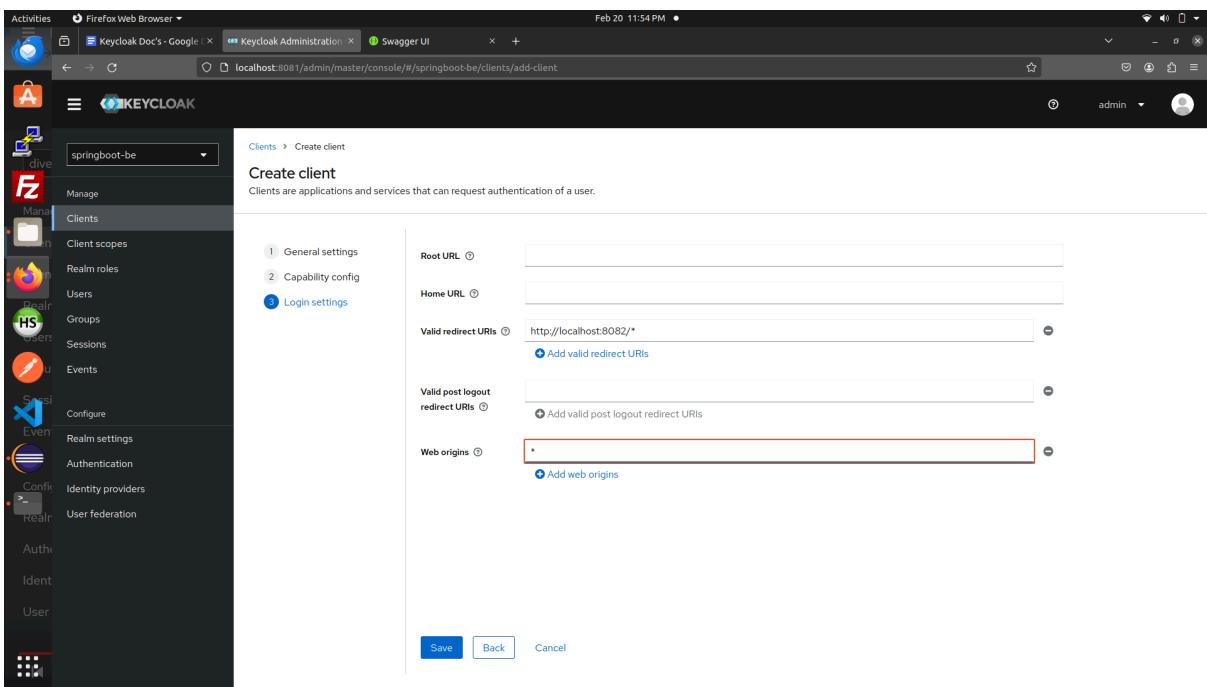
**EG: one origination(REALM) can be but different sectors(CLIENT).**

## Login in one App:

Create a client in the Realm.



The screenshot shows the Keycloak Administration interface in a Firefox browser. The left sidebar is titled 'master' and includes options like Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The main content area is titled 'Clients' and contains a table of existing clients. The table columns are Client ID, Name, Type, Description, and Home URL. The clients listed are: account, account-console, admin-cli, broker, just-realm, master-realm, security-admin-console, and springboot-be-realm. The 'account' client has a Home URL of <http://localhost:8081/realm/master/account>. The 'account-console' client has a Home URL of <http://localhost:8081/realm/master/account/>.



The screenshot shows the 'Create client' form in the Keycloak Administration UI. The left sidebar is titled 'springboot-be' and includes the same set of management options as the first screenshot. The main form is titled 'Create client' and contains tabs for General settings, Capability config, and Login settings. Under 'General settings', there are fields for Root URL (empty), Home URL (empty), Valid redirect URIs (containing [http://localhost:8082/\\*](http://localhost:8082/*)), and Valid post logout redirect URIs (empty). Under 'Web origins', there is a single entry: [\\*](#). At the bottom of the form are 'Save', 'Back', and 'Cancel' buttons.

The screenshot shows the Keycloak Administration UI in a Firefox browser. The left sidebar is dark-themed and includes icons for Identity providers, Authentication, Configuration, and various realm settings. The main navigation bar has tabs for 'Keycloak Administration' and 'Swagger UI'. The current page is 'localhost:8081/admin/master/console/#/springboot-be/clients/aa1adf7d-b06e-418c-93bc-edc3ab0d0e07/settings'. A success message 'Client created successfully' is displayed in a modal. The page contains sections for 'Access settings' (Root URL, Home URL, Valid redirect URIs, Valid post logout redirect URIs, Web origins, Admin URL) and 'Capability config' (Client authentication On, Authorization Off, Authentication flow: Standard flow, Direct access grants, Implicit flow, Service accounts roles). Buttons for 'Save' and 'Revert' are at the bottom.

## In Authentication:

The screenshot shows the Keycloak Administration UI in a Firefox browser. The left sidebar is dark-themed and includes icons for Identity providers, Authentication, Configuration, and various realm settings. The main navigation bar has tabs for 'Keycloak Administration' and 'Swagger UI'. The current page is 'localhost:8081/admin/master/console/#/springboot-be/clients/aa1adf7d-b06e-418c-93bc-edc3ab0d0e07/permissions'. A success message 'Successfully created the permission' is displayed in a modal. The page shows a form for creating a new permission: 'Name' (Test permission), 'Description' (empty), 'Apply to resource type' (Off), 'Resources' (Test Resource), 'Policies' (Test Policy), and 'Decision strategy' (Unanimous). Buttons for 'Save' and 'Cancel' are at the bottom.

The image displays two screenshots of the Keycloak Administration UI, both taken at 12:00 AM on February 21.

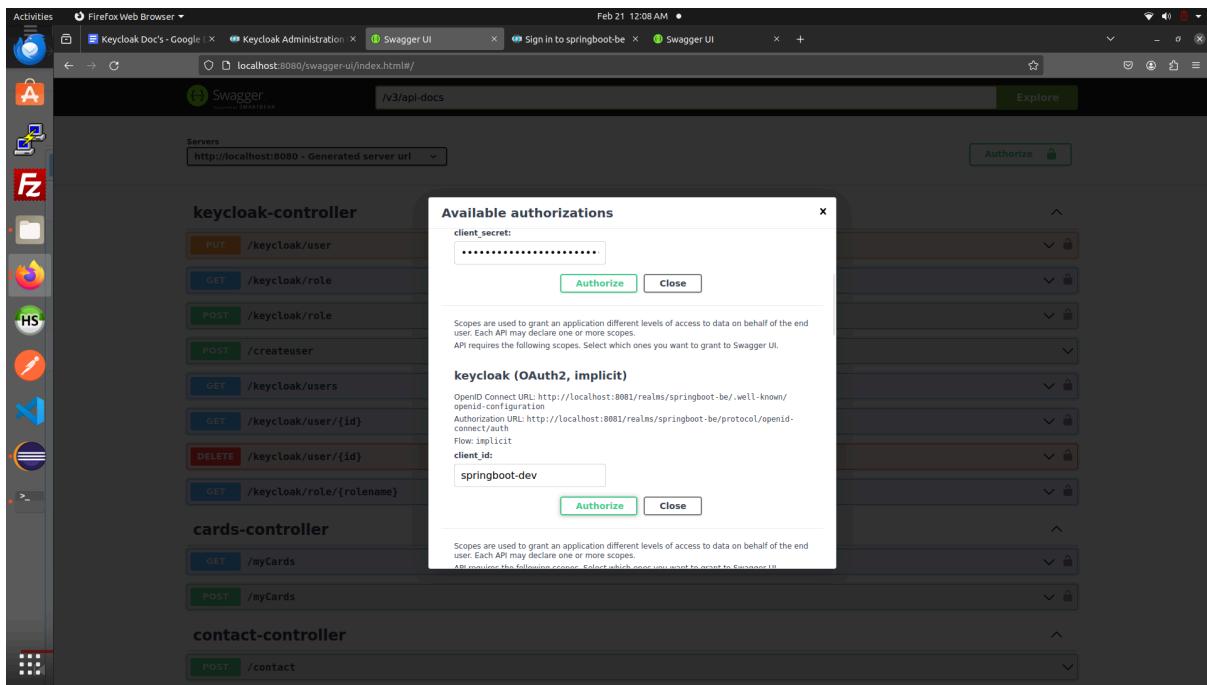
**Screenshot 1: Test Policy**

This screenshot shows the "Test Policy" configuration page. The "Name" field is set to "Test Policy". Under "Roles", there is a single role named "owner" with a checkbox next to it. The "Logic" section is set to "Positive". At the bottom, there are "Save" and "Cancel" buttons.

**Screenshot 2: Test permission**

This screenshot shows the "Test permission" configuration page. The "Name" field is set to "Test permission". The "Apply to resource type" switch is turned off. Under "Resources", "Test Resource" is selected. Under "Policies", "Test Policy" is selected. The "Decision strategy" is set to "Unanimous". At the bottom, there are "Save" and "Cancel" buttons.

By using **implicit Login** we can achieve the SSO.



## To Access ADMIN API for Other User :

(eg: if user login but in some cases users need to access the certain API's so this logic will be used).

This (/keycloak/user/{id}) API needs to be accessed by a certain user.

In Spring boot :

```

5 "credentials": {
6   "secret": "zfopel8nyMgZ4kpUADHxHOn6WI67Zurx"
7 },
8 "http-method-as-scope": true,
9 "paths" : [
10  [
11    "path": "/restaurant/public/list",
12    "enforcement-mode": "DISABLED"
13  ], {
14    "path": "/restaurant/public/menu/*",
15    "enforcement-mode": "DISABLED"
16  }, {
17    "path": "/swagger-ui/*",
18    "enforcement-mode": "DISABLED"
19  }, {
20    "path": "/v3/api-docs/*",
21    "enforcement-mode": "DISABLED"
22  }, {
23    "path": "/keycloak/users/{userId}",
24    "claim-information-point": {
25      "claims": {
26        "uri_claim": "{request.uri}"
27      }
28    }
29  }, {
30    "path": "/keycloak/*",
31    "enforcement-mode": "ENFORCING"
32  }
33 ]
34 ]

```

The /keycloak/user/{id} we enabling for other user to access and we need to set claim-info In the uri\_claim we are setting the param variable,So we used {request.uri}.

If we add the /keycloak/user/{id} reset of the /keycloak API will not work so we will enforce the /keycloak/\*.

## Need to create JavaScript Policy Provider:

Create a Folder to create the Files.

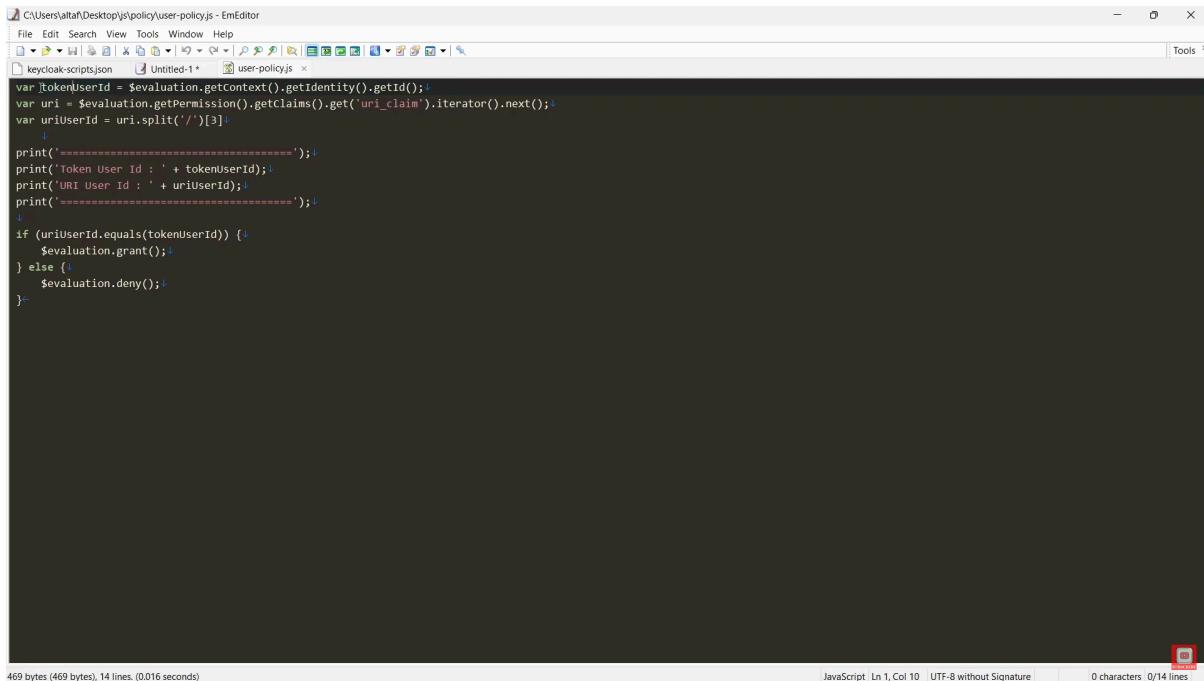
Create Folder name as META-INF , Inside the META-INF keycloak-script.json.

Open the keycloak-script.json file.

```
{  
  "policies": [  
    {  
      "name": "User Policy",  
      "fileName": "user-policy.js",  
      "description": "User specific access policy"  
    }  
  ]  
}
```

The filename is given as use-policy.js.

Go back to from META-INF and create user-policy.js.



```

C:\Users\altaf\Desktop\js\policy\user-policy.js - EmEditor
File Edit Search View Tools Window Help
keycloak-scripts.json Untitled-1 user-policy.js
var tokenUserId = $evaluation.getContext().getIdentity().getUserId();
var uri = $evaluation.getPermission().getClaims().get('uri_claim').iterator().next();
var uriUserId = uri.split('/')[3];
print('=====');
print('Token User Id : ' + tokenUserId);
print('URI User Id : ' + uriUserId);
print('=====');
if (uriUserId.equals(tokenUserId)) {
    $evaluation.grant();
} else {
    $evaluation.deny();
}

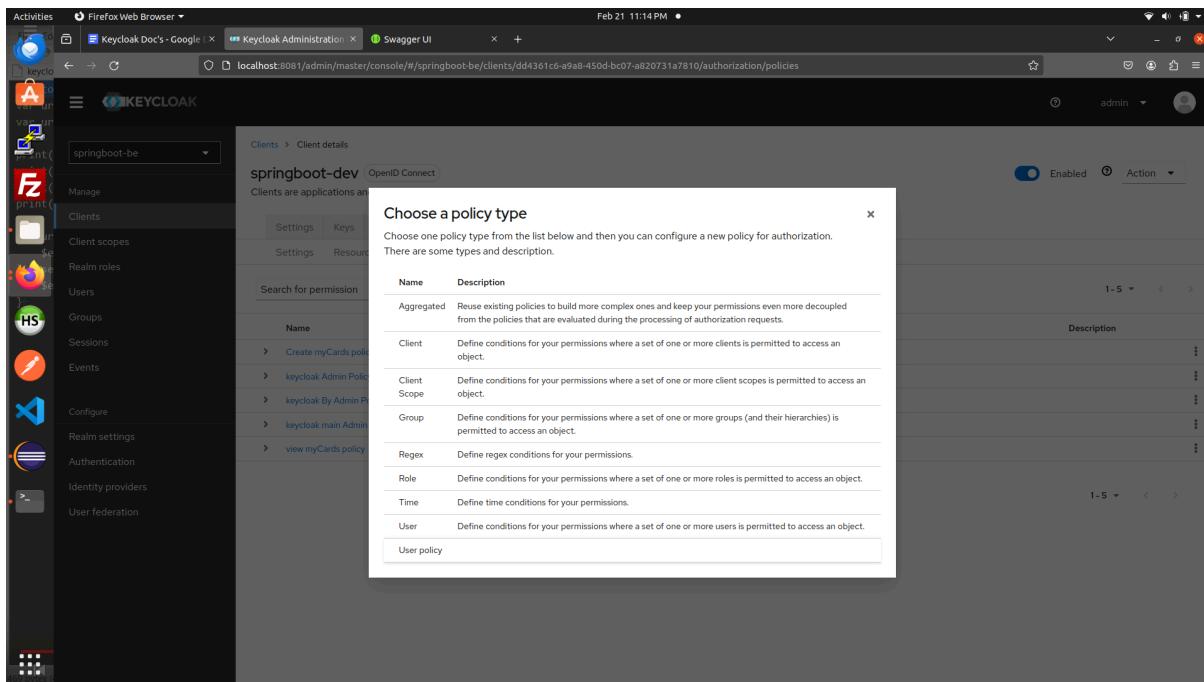
```

469 bytes (469 bytes), 14 lines. (0.016 seconds)    JavaScript | Ln 1, Col 10 | UTF-8 without Signature | 0 characters | 0/14 lines

In the uri we are getting an uri-claim which we will mention in the Spring-boot json file.  
 Select the All folder and file compare to zip and change the exe to Zip to jar so jar file will be created.

Then copy the jar and place it in the /keycloak/providers.

Go -> claim -> Authentication-> Policies -> create policy you can see the User-policy.



Choose a policy type

Choose one policy type from the list below and then you can configure a new policy for authorization. There are some types and description.

Name	Description
Aggregated	Reuse existing policies to build more complex ones and keep your permissions even more decoupled from the policies that are evaluated during the processing of authorization requests.
Client	Define conditions for your permissions where a set of one or more clients is permitted to access an object.
Client Scope	Define conditions for your permissions where a set of one or more client scopes is permitted to access an object.
Group	Define conditions for your permissions where a set of one or more groups (and their hierarchies) is permitted to access an object.
Regex	Define regex conditions for your permissions.
Role	Define conditions for your permissions where a set of one or more roles is permitted to access an object.
Time	Define time conditions for your permissions.
User	Define conditions for your permissions where a set of one or more users is permitted to access an object.
User policy	

## Create a Resource :

The screenshot shows the Keycloak Administration interface in a Firefox browser. The left sidebar is dark-themed and shows a navigation tree with 'Clients' selected. The main content area is titled 'User Resource' under 'Client details'. The form fields are as follows:

- Owner:** springboot-dev
- Name:** User Resource
- Type:** (empty)
- URIs:** /keycloak/user/[id]
- Authorization scopes:** GET
- Icon URI:** (empty)
- User-Managed access enabled:** Off

Below the form, a note says: "No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair." A blue link "Add an attribute" is present.

At the bottom are 'Save' and 'Cancel' buttons.

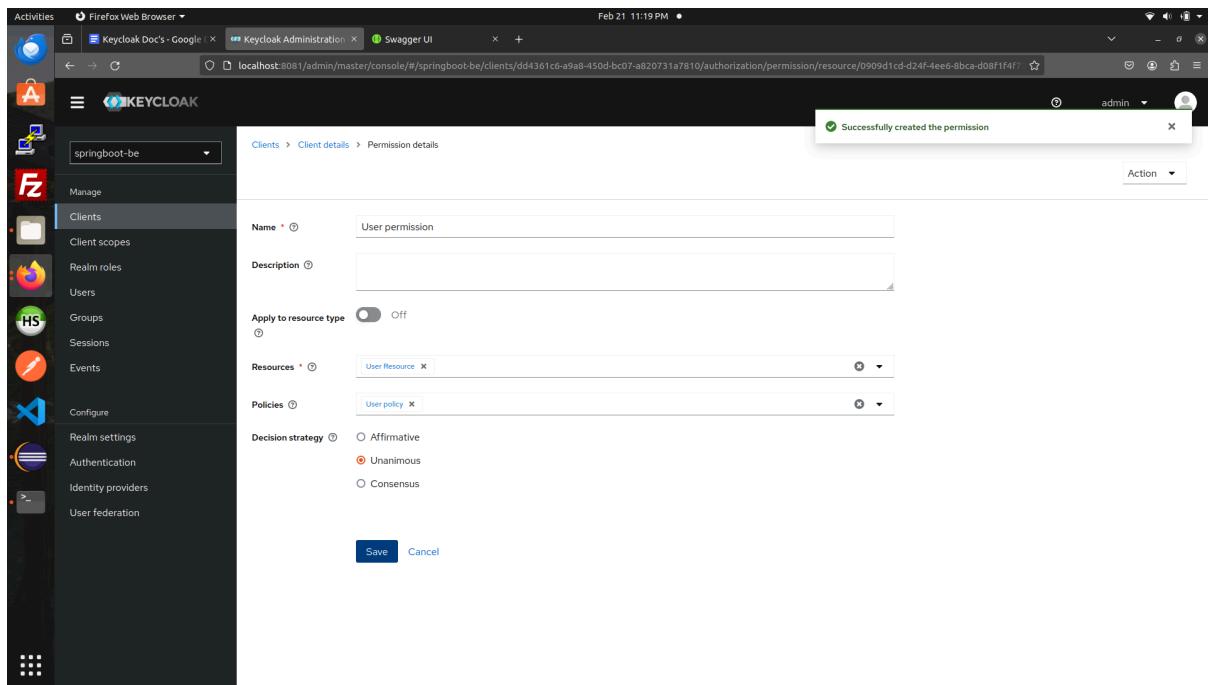
## Create Policy (select the user-policy)

The screenshot shows the Keycloak Administration interface in a Firefox browser. The left sidebar is dark-themed and shows a navigation tree with 'Clients' selected. The main content area is titled 'Create script-user-policy.js policy' under 'Client details'. The form fields are as follows:

- Name:** User policy
- Description:** (empty)
- Code:** 1

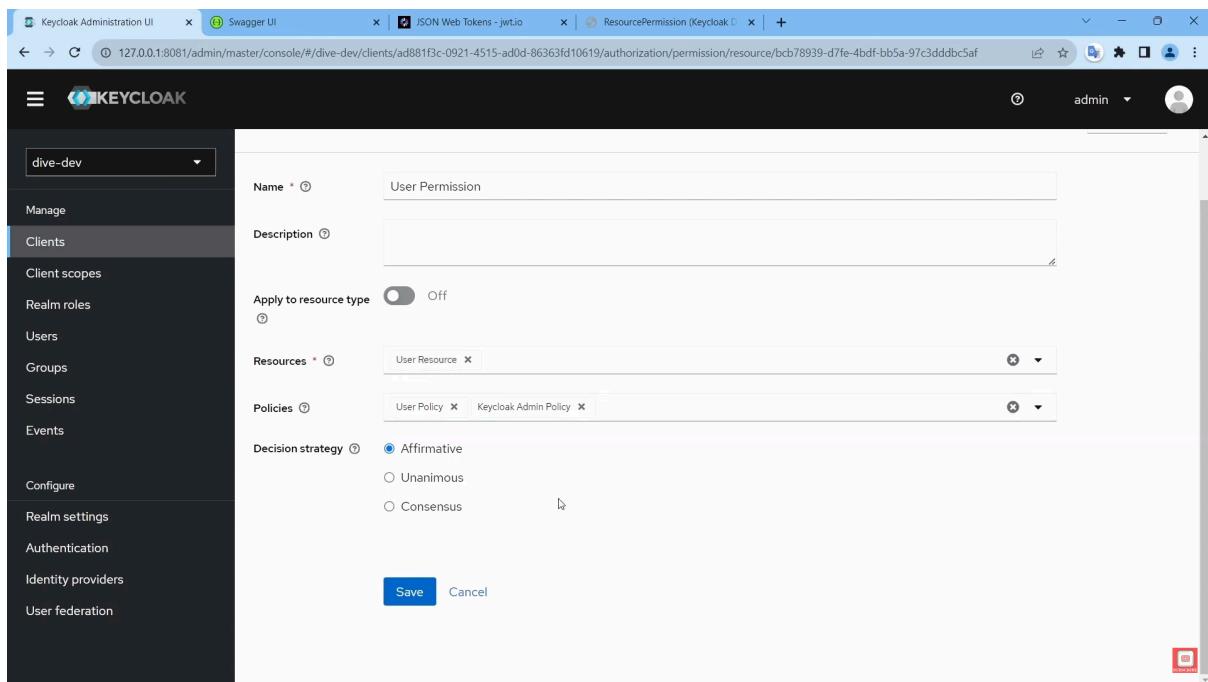
A large text area labeled 'Code' contains the number '1'. At the bottom are 'Save' and 'Cancel' buttons.

## Create Permission:



If we test the API's It will work for the same user Login and same UserID need to be equal then they can use the API.

If Admin needs access we can specify in the Permission.

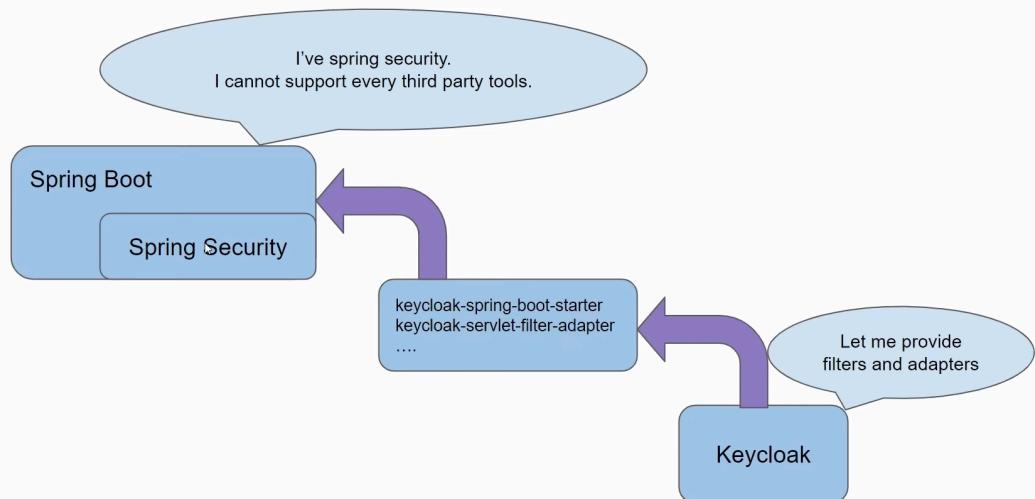


If we add More than Policy we need to concentrate on Decision strategy.

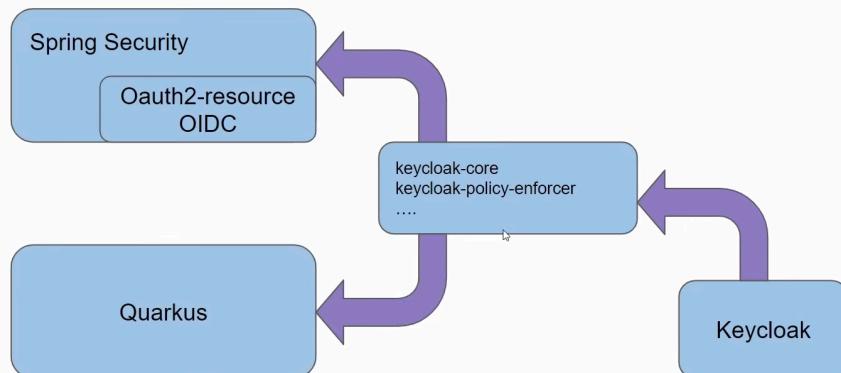
Affirmative : Atleast one of Policy need to be satisfied.

Unanimous : All the Policy needs to be satisfied.

## Keycloak deprecates adapter libraries from version 22



## Keycloak deprecates adapter libraries from version 22



We can add Extra CLAIMS using spring boot:

[View in github:](#)